(1)

September 1991

MTR11259

Marian J. Murphy

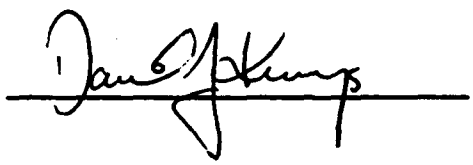
USI Rapid Prototyping Tool Evaluations Survey

CONTRACT SPONSOR N/A
CONTRACT NO. N/A
PROJECT NO. D40G
DEPT. D042

MITRE

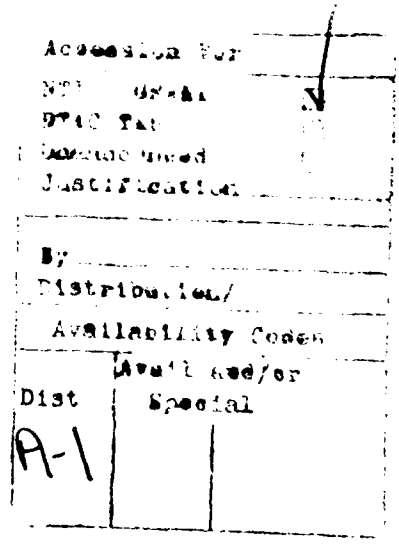
The MITRE Corporation
Bedford, Massachusetts

Department Approval: 

MITRE Project Approval: 

ABSTRACT

The Human Factors Engineering for User System Interface (HFE/USI) Specialty Group has conducted several evaluations of rapid prototyping tools to support the design and development of user interfaces for command and control systems. To standardize the evaluation methodology, we compiled a list of command and control criteria and rated tools against them. These ratings can be used to select a tool that will closely match the needs of a MITRE/ESD program. This paper describes the evaluation methodology and applies it to the review of the following tools: VAPS, LUIS/SMS, SL-GMS, DataViews, TAE Plus, and SET.



ACKNOWLEDGEMENTS

I would like to thank S. E. Campbell and R. C. Couture for their contributions to the tool reviews, and J. N. Mosier for her advice with the evaluation criteria. I would also like to thank R. Coltman and M. E. McBournie for their invaluable help.

EXECUTIVE SUMMARY

INTRODUCTION

There is a growing interest in the use of rapid prototyping for the design and development stages of government/military system acquisition. Rapid prototyping is a process of iteratively refining a system based upon the user's feedback. The prototype can emulate limited functionality of the "real" system, or it can be developed into a deliverable system itself.

Prototyping is particularly useful in the design and development of user-system interfaces (USIs). As the system's most visible part, the user interface can have a great effect upon the user's acceptance of the system. Use of USI rapid prototyping tools is the easiest and quickest approach to user interface design and development. These tools encourage experimentation by maintaining flexibility of design in the development of graphic and text-based user interfaces. A prototype of the user interface provides the user with a set of functional requirements that can be seen and, if not optimal, modified without risking lengthier development.

The Human Factors Engineering for User System Interface (HFE/USI) Specialty Group has conducted evaluations of USI rapid prototyping software tools to support MITRE/ESD projects. There are many commercial-off-the-shelf (COTS) rapid prototyping tools on the market today, and it can be an overwhelming task to sift through all the literature. To make tool selection easier, we are presenting our evaluation results for the following tools:

- VAPS by Virtual Prototypes
- Lockheed User Interface System and Softcopy Mapping System (LUIS/SMS)
- Graphical Modeling System by SL Corporation (SL-GMS)
- DataViews by V.I. Corporation
- Transportable Applications Environment (TAE) Plus by NASA/Goddard Space Flight Center
- Software Engineering Toolkit (SET) by CASET Corporation

EVALUATION METHODOLOGY

To standardize the evaluation of USI rapid prototyping tools, and to provide a basis for comparison, we have developed a list of evaluation criteria based upon typical government and military requirements. We have acquired the tools and used them to implement model command and control application user interfaces. Many of the tools were acquired permanently and have been used in our support of MITRE/ESD programs. We have evaluated each tool against the list of criteria and developed the ratings into a set of "tools-at-a-glance" tables. If a new tool version was released after our evaluation, we presented the new release's modifications and enhancements in the review. We also generated an overview description and a summary of evaluation results for each tool.

EVALUATION CRITERIA

The evaluation criteria against which the USI rapid prototyping tools are evaluated fall into five distinct categories.

- **Capabilities** refer to both graphic and general utility functions such as geographical map generation, zooming and panning, labeling, text handling, display objects, windowing, and code generation.
- **Usability** issues include such topics as direct manipulation, menuing, precise object placement, environment flexibility, and response time. The amount of programming required to develop a prototype, the learning curve involved in using the tool, and the quality of the documentation are also discussed.
- **Quality of the vendor support** encompasses different types of assistance throughout the development process, including installation, training, and telephone assistance.
- **Hardware** issues involve such topics as display devices, input devices, computing platforms, and portability.
- **Software** issues deal with adherence to industry standards and compatibility with other commercial software tools.

VAPS

Manufacturer:	Virtual Prototypes, Inc., Montreal, Canada
Versions:	1.04, 2.05
Platforms:	Silicon Graphics, Sun 4, and DECstation
Cost:	$15,000 to $41,000, depending on the configuration

The VAPS Prototyping System is a high-end USI rapid prototyping tool. VAPS provides a flexible prototyping environment in which the user interface can be modified quickly and easily. It consists of four software subsystems that collectively provide all the capabilities needed to quickly implement a user interface: graphics editor, integration menu, logic editor, and runtime manager.

Although VAPS was originally developed to support avionics applications, it can also support a range of user interface applications. It is particularly useful for the prototyping of hardware devices such as equipment and display consoles. It has been used successfully to prototype control panels and transceivers for a tactical system, command center displays, instrumentation panels, tactical radio panels, and aircraft cockpits.

Evaluation Summary

Because its emphasis is on objects such as lights, buttons, knobs, and dials, VAPS is well-suited to the modeling of hardware devices. A user input logging capability has been added to the record the user's actions. Another major strength of VAPS is its ability to allow

a developer to quickly and easily build and modify a user interface. It is a powerful USI rapid prototyping tool that can significantly reduce the development time of a prototype in comparison to many other commercial rapid prototyping tools. However, VAPS code should be optimized to minimize overhead and maximize response time. Because of its overhead requirements, it is only feasible to run VAPS on high performance workstations, such as the Silicon Graphics workstation.

LUIS/SMS

Manufacturer: Lockheed Missiles and Space Company, Inc., Austin, Texas
Versions: LUIS 6.33b, II; and SMS 6.63
Platforms: DECstation, HP/Apollo, Silicon Graphics, and Sun 4
Cost: $50,000 for both LUIS and SMS

LUIS/SMS consists of an integrated environment of two systems. The first is the Lockheed User Interface System (LUIS), which provides a deeply nested menuing system to support development of a user interface. The second is the Softcopy Mapping System (SMS), which supports development of geographical mapping systems. Although LUIS and SMS can both run as stand-alone systems, they were developed to run together. LUIS provides the interface through which the user can interact with the mapping system. Also, without the advantages of SMS' special mapping capabilities, there are many user interface development systems that are easier to use than LUIS. As a package, however, LUIS/SMS provides a unique set of capabilities. The newest version, LUIS II, is currently being distributed.

LUIS/SMS was designed by Lockheed to support military applications that involve geographical maps. It can support many different levels of map detail, and it provides mechanisms for ensuring optimal performance.

Evaluation Summary

The strengths of the integrated LUIS/SMS package are numerous. First, LUIS enables the developer to create the user interface without programming. Through a complex menuing system, the developer can create both the "look" and the "feel" of the user interface independent of any coding. The menus allow the developer to specify an action to be executed when a button is selected or deselected. LUIS, in fact, requires coding later in the development process than do many other rapid prototyping tools. Coding is necessary to bind in application programs, including SMS maps, to the LUIS interface.

Another major strength of LUIS/SMS is that it combines the capabilities of user interface design with map display generation. This unique combination is very useful for command, control, communications and intelligence (C3I) applications. LUIS/SMS also provides an input logging feature that is useful for performing usability testing on the user interfaces developed with LUIS/SMS. Finally, LUIS/SMS runs on a range of hardware platforms.

Tradeoffs are, of course, involved in many of these capabilities. For example, the same complexity of the menuing system that allows the developer to create the "look and

feel" of the interface independent of programming also can cause the tool's interface to seem complicated and cumbersome. Also, although programming is not required quite as early in the development process as with many other tools, the integrated LUIS/SMS does, in fact, require a substantial amount of coding to create a mapping application. It includes over 300 high-level geographical mapping, display object manipulation and inter-process communication (between LUIS and external applications, including SMS applications) programming routines. There is a steep learning curve associated with the tool.

SL-GMS

Manufacturer: SL Corporation, Corte Madera, California
Versions: 3.0f1, 4.0
Platforms: Sun, Megatek, Silicon Graphics, VAXstation, HP 9000/3000/Apollo
Cost: $12,000 for a development license;
$2,500 for a runtime license

The Graphical Modeling System (SL-GMS) is a rapid USI prototyping tool that supports easy generation of graphic objects and assignment of dynamics to them. SL-GMS consists of two main components. First, it provides an easy-to-use interactive graphics editor called "SL-Draw," which supports development of dynamic graphic displays and objects. Second, SL-GMS provides a function library called "SL-GMF" whose more than 500 routines can be embedded within a C, Fortran, Ada, or Pascal control program. "Control program" is the term we will use to refer to the program that "controls" the functionality of a prototype, such as the sequencing of displays, the appearance of a pop-up window, or the reading of text input from the end user.

SL-GMS was originally developed to prototype process control within a manufacturing environment, but it has evolved into a general purpose tool.

Evaluation Summary

SL-GMS is relatively inexpensive compared to some of the higher-end tools, and yet it is a good general purpose tool. It allows the "look" of the prototype to be free from all graphics coding. Screen development is easy and flexible. However, because display sequencing cannot be defined in SL-Draw, SL-GMS does require programming early in the development process. While the learning curve associated with SL-GMS can be significant, it is not as steep as those of some more powerful tools. SL-GMS also minimizes overhead, freeing the developer from worry over performance. SL-GMS is a tool well-suited to interfaces that require simple display objects, such as buttons and text input fields, and do not require complex geographical maps.

TAE PLUS

Manufacturer:	NASA/Goddard Space Flight Center ; distributed by COSMIC, NASA's software distribution center located at the University of Georgia
Versions:	4.1, 5.1
Platforms:	Sun 3/4, Dec, HP/Apollo workstations
Cost:	$500 for the general public, $250 for the government for 4.1; $750 and $350 respectively for 5.1

TAE Plus is a portable, X Window-based software package that supports design and development of interactive graphic user interface application systems. TAE Plus consists of two main subsystems: the Workbench and the Window Programming Tools (WPT). The Workbench is the primary tool for user interface development. It can be used to interactively create or edit the graphic portion of the interface and generate an application program. The WPT library of programming routines handles the functionality of the interface. TAE also provides a graphics editor called "Idraw," but it is only used for creating and customizing special object types; all the interface development is done in the Workbench. Version 5.1 was released in May 1991. TAE Plus is based on the X Window System, X11 Release 3.

Evaluation Summary

TAE Plus is a cost-effective user interface development tool. It is particularly useful for developing user interfaces that require a wide variety of interactive display objects and have a fairly well-defined interface design. Its development environment is limited in flexibility, but it provides an easy interface to the X Window System. (Other X-based user interface development tools include Builder Xcessory, XBUILD and UIMX.)

TAE allows the developer to build a user interface under X Windows without requiring an understanding of the complexities of X. However, TAE allows the developer to access the X toolkit and X library directly if necessary.

TAE Plus supplies a wide range of ready-made, high-level display objects such as pull-down menus, check boxes, radio buttons, and scroll bars. The objects in 4.1 are based on MIT's Athena widgets; and the objects in 5.1 are based on MOTIF. Therefore, it is very useful for user interfaces that require many different interaction styles.

Perhaps TAE Plus' most important strength is its price. It offers a rather complete set of high-level display objects at a fraction of the cost of the other tools discussed in this paper.

DataViews

Manufacturer:	V.I. Corporation, Northhampton, Massachusetts
Versions:	7.0, 8.0, 9.0
Platforms:	Sun, HP/Apollo, Data General, IBM RS/6000, VAXstation, DECstation, and Silicon Graphics
Cost:	$17,000

DataViews is a product of V.I. Corporation. It is a general purpose USI rapid prototyping tool that allows the developer to build a dynamic graphic user interface. DataViews consists of two major components: the graphics editor DV-Draw, and the programming library DV-Tools. DV-Draw is a menu-driven, direct manipulation graphics editor that allows displays to be created, animated, and executed independent of coding. DV-Tools is an extensive library of routines callable from user-supplied "C" code that support the dynamics and functionality of the interface.

Like SL-GMS, DataViews was originally developed to support process control manufacturing applications, but has become a more general purpose tool.

Evaluation Summary

DataViews is relatively affordable. It is particularly useful for prototypes that require a great deal of dynamic graphics. DataViews provides a good graphics editor that allows both the "look and feel" of the prototype to be defined without programming. It also provides a preview function that allows the developer to test the functionality of the prototype within the graphics editor. Its dynamic objects can be driven by any type of data source, and they can be defined to move across the screen with real-time positioning. However, the number of items that can move on the screen at one time is limited.

SET

Manufacturer:	CASET Corporation, San Juan Capistrano, California
Versions:	3.4, 3.8
Platforms:	Sun, HP/Apollo, DEC VAX, DECstation and VAXstation, Silicon Graphics and the PRIME 50 Series
Cost:	$7,500 for all five modules; $2,000 for each individual module

SET is useful for developing prototypes that are primarily text-based and require a minimal selection of display objects. It provides a suite of programming libraries, and it is a programming-intensive tool. SET has undergone a fairly extensive upgrade since it was first released, and version 3.8 has become a markedly better user interface design tool than the original release of SET. Although it does not directly support geographical mapping, a new module that provides raster and vector graphics capabilities became available September 1991. Also, version 3.8 is compatible with the X Winclow system and Motif.

Evaluation Summary

SET version 3.4 is a useful tool for expert developers who would rather develop a prototype through high-level programming with the routine libraries than through use of a menuing system or direct manipulation. Unlike many of the other tools, SET has undergone an extensive set of modifications during the last two releases. An interactive graphics editor was added; the documentation was improved; and the tool was made X/Motif compatible. Use of the libraries can be a significant improvement over coding from low-level graphics packages, such as CORE or GKS. However, the SET programming libraries can present a

significant learning curve even for expert programmers, simply because of the poor documentation.

CONCLUSIONS AND RECOMMENDATIONS

There is no one "best" USI rapid prototyping tool. Selection of a tool must be based upon the particular needs of a given system. Each of the tools has its own strengths and weaknesses, and these should be fitted to the system requirements. For example, if a prototype requires high-level display objects such as scrolling text windows and pull-down menus superimposed on complex geographical map data, LUIS/SMS would be preferable. If, on the other hand, the prototype is to be a simulation of an Air Force display console, complete with dials and buttons, VAPS would be the appropriate tool. DataViews and SL-GMS might be considered for the development of a more general purpose prototype, such as different applications of computer information systems. TAE Plus might be chosen for a prototype that requires many high-level display objects such as radio buttons, check boxes, and scroll bars, but has a limited budget. Beyond functional requirements, it is important to consider skills and training. For example, developers who have a great deal of experience with software development may choose to use SET, which is very programming-oriented.

The "Tools-At-A-Glance" tables (appendix) can be used to determine the tool best suited to an application. To do so, the user can select a subset of pertinent criteria and pick the tool that was rated highest against them. It may also be beneficial to use the ratings in a weighted factor analysis to determine which tool to select. To use this approach can assign numerical weights to the criteria, according to importance. Numerical values can also be assigned to the rating: three, two and one for check plus, check, and check minus, respectively. Next, multiply the numerical rating scores by each weight factor, and add the total for each tool. The tool with the highest total score is the "best" match for the application.

Similarly, the evaluation criteria can be used as a basis for other evaluations. To conduct an evaluation of a tool not included in this paper, compile a list of pertinent criteria. Next, design a model benchmark prototype that incorporates all these criteria. Finally, use the tool to implement the benchmark application. In this way, the degree to which the tool supports each of the criteria can be rated.

In conclusion, USI rapid prototyping tools can be critical to the development of high-quality user interfaces. Prototyping lifts a design from paper and allows the user to physically "see and feel" the system specifications. Prototyping tools provide the system developer with the power and flexibility to experiment with design options. Because these tools are often expensive, MITRE/ESD projects may want to delay purchasing one until they are confident it suits their needs.

The HFE/USI Specialty Group can assist in this effort. We have many of the current USI prototyping tools in-house. We are prepared to present demonstrations of our in-house tools. We also evaluate new tools upon request. As new USI rapid prototyping tools emerge on the market, we will continue to evaluate them and update future editions of this paper with the results.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION ONE

BACKGROUND

1.1 PURPOSE

Over the past decade, there has been a growing interest in the use of rapid prototyping for the design and development stages of system acquisition. Because rapid prototyping involves iterative refinement of a system based on feedback from the user, it can align a system closely to user specifications, and improve user acceptance of the final system.

Prototyping is particularly useful in the design and development of user interfaces. The interface between the user and the system is the system's most visible part, and so has a great effect upon user acceptance. A prototype of the user interface provides the user with a set of functional requirements that can be seen and refined.

Use of rapid prototyping tools is the easiest and quickest approach to designing and developing user interfaces. These tools encourage experimentation by maintaining flexibility of design in the development of graphic and text-based user interfaces. MITRE's Human Factors Engineering for User System Interface (HFE/USI) Specialty Group has been involved in the evaluation of USI rapid prototyping software tools to support MITRE/ESD projects. There are many commercial-off-the-shelf (COTS) rapid prototyping tools on the market today, and it can be an overwhelming task to sift through all the literature. To make tool evaluation easier, we report here our evaluations of six USI rapid prototyping tools:

- VAPS by Virtual Prototypes
- Lockheed User Interface System and Softcopy Mapping System (LUIS/SMS)
- Graphical Modeling System by SL Corporation (SL-GMS)
- DataViews by V.I. Corporation
- Transportable Applications Environment (TAE) Plus by NASA/Goddard Space Flight Center
- Software Engineering Toolkit (SET) by CASET Corporation

The results can be used by MITRE/ESD projects in selecting tools to fit their needs.

1.2 METHODOLOGY

To standardize evaluation of the user interface rapid prototyping tools and to provide a basis for comparing them, we developed a list of evaluation criteria. We surveyed various government and military applications to determine common functionality requirements and implementation preferences. We defined and then categorized the criteria; the categories include capability, usability, vendor support, and software and hardware issues.

To expedite the process of recording our evaluation results, we developed a set of questions that address the major points of each evaluation criterion. We acquired each of the tools for a trial evaluation period, and used them to build model command and control application user interfaces for a period of at least three months. (For more details on benchmark applications, see "Evaluating Graphical User Interface Tools for Command and Control Applications," Daniel J. Kurys, National Computer Graphics Association Annual Conference, March 1990). Many of the tools were acquired permanently and have been used to support our softshell work for MITRE/ESD programs.

As we used each tool, we evaluated it against the list of criteria. The tools were rated on a three-point scale:

- Directly supports this criterion (check plus)
- Can support the feature but requires additional effort by the developer (check)
- Does not support this criterion at all (check minus)

For criteria such as compatibility with input devices and hardware platforms, the ratings mean the following: "check plus" means the tool can support more than two; "check" means the tool supports two; and "check minus" means it supports only one. These ratings were placed in a set of "tools-at-a-glance" tables. Finally, we generated an overview description and a summary of evaluation results for each tool.

To select a USI rapid prototyping tool, use the following three-step process:

- Identify pertinent criteria and assign them numerical weights according to relative importance
- Evaluate the tools against the criteria
- Develop a decision matrix as shown in section 4

1.3 SCOPE

Section 2 defines and lists questions for each of the evaluation criteria. The questions can support the reader in evaluating a tool not included in this paper. The tool overviews and evaluation results are summarized in section 3. Section 4 reports our conclusions and recommendations based upon the evaluation results. A decision-making chart that can be used for selecting a tool is also shown. The "tools-at-a-glance" tables are contained in the appendix.

SECTION TWO

EVALUATION CRITERIA

2.1 INTRODUCTION

The criteria against which the tools are evaluated fall into five distinct categories: capability, usability, vendor support, hardware issues, and software issues.

Capabilities include both graphic and general utility functions that can assist in the prototyping process. Examples are geographical map generation, zooming and panning, labeling, text handling, display objects, windowing, and code generation.

Usability issues include such topics as direct manipulation, menuing, precise object placement, environment flexibility, and response time for the tool itself. The amount of programming required to develop a prototype, the learning curve involved in using the tool, and the quality of the documentation are also discussed.

Quality of the vendor support is a third key issue in the evaluation of a rapid prototyping tool. It can encompass different types of assistance throughout the life cycle of the development process, including installation, training, and telephone assistance.

Hardware issues include topics such as display devices, input devices, computing platforms, and portability.

Software issues deal with concerns such as adherence to industry standards and compatibility with other commercial software tools.

Before proceeding to individual tool evaluations, we will examine each of these categories in detail, and pose the questions that should be asked as part of the evaluation process.

2.2 CAPABILITY ISSUES

2.2.1 Geographical Mapping

Since mapping capabilities are an important part of many C2 displays, it is often important to be able to turn geographical maps into prototypes. Rapid prototyping tools can support mapping capabilities at two different levels. They can directly support development of vector-drawn maps or import maps from outside sources. A few rapid prototyping tools directly support development of geographical mapping systems. These tools accept input data and create complex geographical maps from it. Many tools provide for importing maps from external sources.

Evaluation Questions:

- Does the tool support a wide variety of available vector and raster map formats (World Database I, II, DTED, DFAD, SPOT, etc.)?
- If so, was it a difficult process?

2.2.2 Graphics Manipulation

The manipulation of graphics includes functions such as zooming, panning, and resetting. Rapid prototyping tools can support a variety of zooming factors. The zooming factors control to what degree the graphics enlarge or shrink each time the "zoom in" or "zoom out" function is invoked. Panning is also assigned a factor. Panning refers to the ability to move a graphic display from the current position to the right, left, up, or down. The amount of movement relative to the graphics is controlled by the panning factor. The reset function undoes any zooming or panning that has been done, returning the graphics to the previous position. These types of manipulations are useful for maps as well as other types of graphics.

Evaluation Questions:

- Does the tool support zooming, panning and resetting graphics?
- Did you find these functions easy to implement?

2.2.3 Toggle Overlays

Toggle overlays allow each level of graphics to be superimposed on the previous level. For example, in a mapping application, the picture may begin with a simple depiction of the country borders. Major roads, rivers, cities, terrain shading, and elevation shading can then each be added to form a composite picture. Toggle overlays permit each separate part of the picture to be toggled on or off independently, improving the tool's performance. Thus toggling off the city names would not require that the total picture be redrawn.

Evaluation Questions:

- Did the tool provide for management of graphic overlays?
- Did the toggle overlays meet your needs for manipulating different types of data?

2.2.4 Labels

Labels include pictures as well as alphanumerics. Both types of labels may be used as part of selectable display objects such as buttons and pull-down menus. Alphanumeric labels are frequently used in command and control user interfaces. Examples of picture labels are the folder, file, and trash can icons commonly used in word processing tools. They are also called "icons."

A useful feature causes labels to reverse video as the cursor moves over them; for example, each option button within a pull-down menu reverses video as the cursor passes over it, and then returns to its original appearance when the cursor is gone. Alphanumeric labels can

also be used as static (non-selectable) labels. For example, a static label may be used to display such information as "ALERT" or "DEFCON 3". It is often useful to be able to define these static labels to flash on the screen. Typical adjustable attributes for alphanumeric labels include font size and style.

Evaluation Questions:

- Does the tool support alphanumeric and picture labels?
- Was it difficult to cause the labels to reverse video as the cursor passed over them?
- Was it difficult to cause static text warnings to flash?
- Does the tool support development of specialized symbology?

2.2.5 Text Handling

Text editing capabilities deal with the usability of a particular high-level display object called a "text field." Text fields are areas on the screen within which the user can input, and the system can output, single or multiple lines of text. The fields can be formatted or unformatted. An example of a formatted field is "--/--/--" in which the user is prompted by the formatting to input a date.

It is useful for a prototyping tool to allow default values to be defined for text fields. Default values can simplify the user's workload. They can also be used as an example of the type of data that should be entered into a particular field.

Navigation between text fields may be done by a pointing device or keyboard. To select a text field, the mouse cursor may simply be placed within the field, or the mouse button may need to be clicked while the cursor is within it. Typically, a special "insertion" cursor indicates the point at which the text will be inserted.

Evaluation Questions:

- Does the tool supply text handling capabilities?
- If so, can the text fields be used for input and output?
- Can they be formatted?
- Was it easy to navigate from text field to text field?

2.2.6 High-level Display Objects

There is a range of high-level display objects that a rapid prototyping tool may or may not support. A "high-level" object refers to a composite object such as a selectable button, as opposed to primitive objects such as the rectangle and text which may comprise it. Direct support of a display object means that the tool provides the object ready-made to the prototype developer rather than providing the primitive objects with which to build it. Typical high-level display objects are selectable buttons, radio buttons, check boxes, labels, text, pull-down menus, pop-up windows and scroll bars.

High-level display objects can greatly simplify the prototype development process. They are most often provided to the prototype developer as either selectable buttons within a graphics editor, or programming calls within a programming routine library.

Evaluation Questions:

- What high-level display objects does the tool directly support?
- What primitive objects does it support?
- Through what mechanism does the tool provide you with these objects?
- Can you modify these "ready-made" objects?
- If so, to what degree?
- Does it provide the high-level objects needed in your application?

2.2.7 Windowing

Windowing capabilities can be useful during development and runtime. They can allow the prototype developer to make modifications to the prototype in one window while watching it run in another window. They can also help a prototype to use screen "real estate" efficiently. Windowing should be accompanied by convenient windowing controls, such as resize, relocate, minimize or maximize, raise or lower, and scrolling capabilities. These types of controls are usually accessible through a menuing system. Resize allows the size of the window to be adjusted. Relocate allows the window to be placed anywhere on the screen. Minimize and maximize allow the window to be adjusted automatically to its smallest or largest size. The "raise" capability selects one window to be the active window; usually the user clicks on that window with the mouse and the selected window is automatically placed in front of any other open windows. This is called "raising" the window. "Lowering" a window is placing a window behind other open windows. Scrolling may be available along the vertical or horizontal edge of the window, or along both.

Evaluation Questions:

- Did the tool provide some type of windowing capability?
- If so, did the tool also provide convenient window control features?
- Upon what windowing systems were these features based: X, SunViews, DECWindows?

2.2.8 Code Generation

An application program may be developed automatically by a software tool. When it is, the generated code is based upon information specified in a form rather than by programming. The application program can be generated in various languages, such as C or Fortran which are compiled, or a user interface language (UIL) which is interpreted at runtime. In the case of rapid prototyping, the code that is needed is an application program that will handle the sequencing of the displays. The generated code includes the subroutines and functions that support the interaction of the prototype. The degree to which the application program requires

modification by the user can vary between tools. For example, the code generation process may generate only subroutine headings and perhaps a main routine; the user must then add the executable commands. Or the generated code may include the subroutines and some of the details of the prototype functionality. In that case the user need make only minor modifications.

Evaluation Questions:

- Did the tool contain a code generation feature?
- What type of code did it generate?
- If so, how complete was the code?
- How extensive were the modifications you needed to make to the code?
- Can the code be modified and executed in the development environment?

2.2.9 User Input Logging

Some rapid prototyping tools provide user input logging as a means of recording the user's commands and input device actions. The user log is usually written into a computer file and lists and time-stamps each entry. User inputs can include any action accomplished through any input device, such as a keyboard, mouse, track ball or light pen. The inputs or actions between the user and tool are often referred to as transactions. The recording of the user's interaction with the tool has several purposes. It can be used by the tool user to recall how a particular task was accomplished, or to estimate how long the task took. It can also be used as a means of tracking the mistakes and successes of a subject in a usability study. User logging can be used as a substitute or complement to other means of observation, such as videotaping a subject. The time stamping aspect of the log is of particular importance here. User logging can also sometimes include the ability to play back the user's interactions and watch the system responses.

Evaluation Questions:

- Did the tool provide any user log generation capability?
- Did you find this feature useful?
- What type of transactions were recorded?
- Where and in what format was the data saved?
- Does the format of the recorded data support the use of data analysis tools?
- Were the transactions time-stamped?
- Could you specify the types of events to be logged by the tool?

2.3 USABILITY ISSUES

2.3.1 Graphic Icons

A common feature of both Macintosh and IBM PC software is selectable buttons labelled with pictures instead of words. These graphic icons are part of the desktop metaphor,

picturing folders, files, mail, and trash cans. As such icons become more common to workstation-based rapid prototyping tools, however, the drawing board, the desktop of the graphic artist, is replacing the conventional desktop. The drawing board metaphor offers:

- Graphic objects including primitives such as lines, arcs and circles as well as high-level display objects such as selectable buttons, scroll bars, and pull-down menus.
- Attributes, including color, line width, font style, and size.
- Operations to be performed on the objects: actions such as move, scale, duplicate, and delete.

Evaluation Questions:

- Does the tool use graphic icons?
- Do the pictures include objects, attributes, and operations?
- Do you find these pictures easier to understand than words?

2.3.2 Direct Manipulation

Physical actions, such as mouse movement, are called "direct manipulations" in contrast to entering commands on a keyboard to perform a function. Direct manipulation uses pointing input devices such as the mouse, track ball, or joystick, and is characterized by pick-and-click interfaces. The pointing input devices operate upon labeled icons. While direct manipulation and graphic icons seem most closely interrelated, text-labelled icons are also commonly used in a direct manipulation environment.

Many rapid prototyping tools consist of subsystems or modules, each of which can have a different interaction technique. Evaluation of this particular criterion may therefore apply either to the entire tool, or to modules of the tool.

Evaluation Questions:

- Does the tool use direct manipulation techniques?
- What pointing device did you use?
- Did the tool allow you to use direct manipulation and type in commands, depending on your preference?

2.3.3 Shallow Menuing

Some tools' menus are nested only one or two layers deep. When menuing systems were first introduced, tool developers found that deeply nested menus could provide hundreds of options at once and allow each lower level of menu options to be driven by the user's current menu selection. However, as nested menus became more popular, tool users encountered several disadvantages. They found that navigating through too many levels of menus and choosing from among too many options can be a cumbersome and confusing process. Even

with the use of pointing input devices such as the mouse or light pen, it can be awkward to navigate through a large number of options. They also found that it was easy to get lost within the nested menus.

For these reasons, shallow menuing became preferable. As a rule of thumb, menus should not be more than two or three layers deep, and should not contain more than ten options. One of the most popular approaches to menu "de-layering," which was adopted by both Apple and IBM for their personal computers, is use of menu bars (or "action bars" in IBM terminology) that allow each menu to be attached to a particular menu heading. The menu bars, attached to the top or bottom of the screen, permit the menu options to be separated and categorized.

Evaluation Questions:

- Does the tool have a menu-based interface?
- If so, how deeply nested are the menus?
- Approximately how many options are included in each menu?
- How extensively are the menus used in the interface?
- Are menu bars used?
- Did you find the menuing system difficult to use?
- Were command alternatives offered?

2.3.4 Object Placement

Tools can be distinguished according to their ability to arrange display objects accurately and precisely on the screen. It can be a tedious process to arrange text or other objects along a horizontal or vertical line. For example, in a text-based display, it may be desirable to left-justify all the text along a vertical line. A series of selectable buttons may need to be aligned within an array, or perhaps a scroll bar must be placed exactly along the edge of a pop-up window.

For applications such as these, an object placement mechanism is necessary. The most common approach is to use a grid whose rectangle size can usually be adjusted. The objects are then snapped to the closest grid point. A more advanced approach is to use "precision points" or "sticky points." When within a certain radius of a precision point, the cursor is drawn towards it. The points are located at the corners and sides of all display objects on the screen, making it easy to place objects next to each other precisely. To align objects along a horizontal or vertical line, a reference line with its own precision points can be used.

Evaluation Questions:

- Did the tool provide any technique for easy object placement?
- Did the object placement techniques provided meet the needs of your application?

2.3.5 Environment Flexibility

A rapid prototyping tool should provide a flexible development environment in which the tool user can experiment with various designs. A flexible environment is one in which the user has control over the development process. For example, a flexible tool does not constrain the user to move step-by-step through the development process via input prompts, nor require that a prototype be developed in any particular order. With a flexible tool, the user can build the interface displays before or after developing the application program, assigning dynamic behaviors or functionality to the display objects before or after they are referenced in the application program. Display objects can be created in any order. The order in which they are created does not restrict their placement in front or in back of other objects.

The degree of flexibility needed in a rapid prototyping system is driven by the developer's mental model. In software development, a mental model is characterized by the thought process or frame of reference of the designer(s) and/or developer(s). For example, a software engineer, human factors engineer, and the user can all have different mental models of the same software tool. The software engineer may want to develop in a way that reduces the amount of coding and/or optimizes the performance. The human factors engineer may wish to make the software as usable as possible. The user may want all kinds of fancy features and colors. These are three mental models: the programmer model, the designer model, and the user model. If the programmer's model dominates during system design, the system may be inflexible but have good performance. If the human factors engineer's model dominates, the system may be easy to use but difficult to implement. If the user's model dominates, the software may be flexible but slow. The goal is to find the optimal balance between these three models.

A flexible prototyping environment gives the user a great deal of freedom. However, it also gives him the freedom to make mistakes. Inherent in this model is the responsibility to remember development and modification details. For this reason, it is a good idea to couple flexibility with an intuitive and simple interface. This combination can make the task of prototype development a creative and enjoyable experience rather than an overwhelming task.

Evaluation Questions:

- Do you feel that you were in control of the development process with this tool?
- Can you build your prototype components in any order you wish?

2.3.6 Screen Overview

A variety of features allow the tool user to view the complete screen. A screen overview function is helpful when the user is panning, zooming, or working with only one object at a time. When using these capabilities, it is easy to become "lost" within the display. For example, the graphics editor of a tool may provide a "drawing space" that is bigger than the view window through which the user works. The user may need to pan around the "drawing space" to create a complete display. In some graphics editors, selection of a single object within a display can cause the remainder of the drawing to disappear from the screen.

The most common method of implementing a screen overview is the inset, usually a small rectangle showing a picture of the complete screen. The overview inset can appear anywhere on the screen, but is most often placed at one of the four corners. Among other implementations of the screen overview concept is the fish-eye view in which the area of interest is surrounded by progressively less detailed representations of the rest of the screen. The area of interest may be a zoomed or panned section, or a particular object. The fish-eye view is a bit trickier to implement, and perhaps more difficult to use than standard views, but may become more widely used in time.

Evaluation Questions:

- Does the tool have a screen overview capability?
- If not, did you feel the need for a screen overview capability while using it?
- Did you feel that getting lost within the complete display was a problem?

2.3.7 Response Time

In interactive computer systems that provide on-line communication between the user and the system, the response time is the time delay between when the user submits a request and when the system responds. In rapid prototyping, the response time is the time delay between manipulating an interactive object and occurrence of the specified function. For example, if a button on the screen is responsible for bringing up a pop-up window, the response time is the time delay between the user selecting the button and the pop-up window appearing on the screen. Or the response time may be the time delay involved in sequencing from one display within the interface to the next. In this case, it is the time taken for the current display to disappear and the new one to appear.

A prototype's response time is affected by the "overhead" of a particular tool, the amount of processing and storage space the tool requires. The response time of a prototype built with a tool will also be affected by the complexity of the prototype. Excessive overhead requirements can decrease a tool's performance, yet it is often a characteristic of powerful, easy-to-use tools. For example, a tool that requires no programming or only small amounts of high-level programming will often require a great deal of overhead. A balance between power and performance must therefore be achieved. In some circumstances, code optimization by both the tool developer and the tool user can be helpful in reaching this balance.

Evaluation Questions:

- How do you rate the tool's performance?
- How do you rate the performance of an application built with the tool?
- Does the tool provide response time similar to what would be required in a real application

2.3.8 Confirmation Checks

A great deal of work and time can be lost due to accidental deletion of a file or failure to save recent work; for example, when a user quits the system without saving. Confirmation

checks avoid disastrous situations such as these by making the user rethink his/her action. They usually take the following forms: "Are you sure you want to delete the file? Are you sure you want to overwrite the file? Are you sure you want to close the file without saving? Are you sure you want to end this session?" Although overuse of such checks can be annoying to the expert user, a reasonable number of them can be helpful to both the novice and expert user.

Evaluation Questions:

- Does the tool provide confirmation checks at potentially hazardous actions e.g., delete file, quit without saving?
- Are they placed at critical points?
- What are some examples of confirmation checks within the tool?

2.3.9 File Recovery

File recovery is related to the idea of confirmation checks. While confirmation checks are used to prevent disastrous situations, file recovery is aimed at enabling the user to recover from such a situation. Many examples of file recovery techniques can be found in new and old computing environments. For example, in the TOPS 20 operating system, a deleted file is available for restoration until the user issues the "expunge" command. In the VAX/VMS environment, saving a new version of a file will also cause a backup version of the original file to be generated. A third example, common to many user interface tools, is the "undo" command. This command allows the user to negate the last command. Unfortunately, in many software tools, "undo" does not apply to the user's response to a confirmation check. In general, a rapid prototyping software tool is said to provide file recovery if there is some technique provided to automatically back up or restore the prototype once it is deleted or modified.

Evaluation Questions:

- Does the tool supply any techniques for file recovery?
- If so, when a file is restored, is it in its original format?
- Does the tool provide an "undo" capability?

2.3.10 Development Without Coding

There are two separate issues to be considered within this criterion: development of the "look" of the prototype, and development of its "feel." At a minimum, a rapid prototyping tool should allow the user to develop the "look" of the interface, including the object colors, attributes, and placement, without programming. This is often accomplished through a graphics editor with which the tool user can create the prototype displays via a pick-and-click interface or a menuing system. Such design flexibility in the development environment is an important aspect of rapid prototyping.

The "feel" of the prototype, on the other hand, which involves assigning behavioral characteristics to the objects and defining the sequencing of the displays, is commonly

developed through programming. Therefore, it is a noteworthy characteristic if a tool allows the functioning of a prototype to be developed without coding.

A development environment that does not require programming has several advantages. First, it can make the task of building a prototype simpler and less time-consuming. Second, it can become a flexible environment by eliminating the need for programming changes when modifying the prototype. However, it can also be a hindrance to the development process for an expert software engineer, who would like to type programming commands. In addition, it may also limit the range of behaviors that can be assigned to objects.

Evaluation Questions:

- Could you specify the "look" of the user interface without coding?
- Could you specify the "feel" of the prototype without coding?
- If not, what was the alternative approach?

2.3.11 Novice Versus Expert User Support

Rapid prototyping tools are used by a diverse community of software engineers, graphic artists, human factors engineers, and others. This diverse population has a range of skills, from users proficient with the tool to beginners. Rapid prototyping tools should therefore be capable of supporting both the novice and the expert user.

The common approach to this problem is to supply two types of interfaces for prototype development. For example, a tool may provide both a pick-and-click interface and a command language to support display generation. The pick-and-click interface is often helpful to novice users, while the expert user may find it cumbersome and inhibiting and prefer to type in commands. To support both types of users, a rapid prototyping tool is preferable if it supplies both types of interfaces.

Evaluation Questions:

- Did the tool supply more than one method of user interaction?
- Did the tool provide an easy mechanism for using a combination of interface styles (e.g., menuing, pick-and-click)?

2.3.12 Learning Curve

One of the most critical criteria for rapid prototyping is the learning curve. The purpose of rapid prototyping software tools is to simplify and streamline the prototyping process, quickly producing a visual example of how the system will appear and function. It is therefore important for a rapid prototyping tool to be relatively easy to use. Learning to use it should not require a great deal of time. An unexpectedly steep learning curve can drain funds and hamper productivity.

Many different factors can affect a tool's learning curve. The quality of its user interface can be a major factor. The interface should be as simple, straightforward and intuitive

as possible. The amount of coding required in the development process can also be a factor. Poor documentation can be a source of frustration. A training course can be helpful and significantly reduce training time.

The learning curve is an area in which many of the rapid prototyping tools on the market today need to improve. The problem stems from a need to balance power with simplicity. The tools must be powerful enough to provide an array of graphic objects and functional behaviors. Yet they must be simple enough to allow the user to produce a prototype rapidly. The difficulty of balancing these two goals often produces a tool that does not provide the high-level display objects, such as scroll bars and pull-down menus, or requires two or three months to learn. The optimal balance between power and simplicity is indeed an important and challenging goal.

Evaluation Questions:

- Was the tool easy or difficult to learn to use?
- How long would you estimate it took you to become proficient with the tool?
- Did you attend a training course for the tool?
- If so, do you feel the course reduced your learning curve?

2.3.13 Usable Documentation

As many software tool users will attest, the quality and the content of the documentation can have a significant effect upon a tool's general usability. The term "usability," which is most often applied to a software system, is just as important when applied to the documentation. This is especially true for more complex software tools, such as rapid prototyping packages.

Documentation should be complete, clearly written, well-organized, and above all accurate. A documentation set should include a "How To Get Started" tutorial and a tool overview as well as system, user, and programmer guides and reference manuals. Each new version of the software should be accompanied by an updated set of documentation. In addition, a newsletter discussing recently discovered software bugs and ways to avoid them can be helpful.

Evaluation Questions:

- What types of documentation provided?
- What is your general impression of the tool's documentation?
- Did the tool features perform as advertised?
- Did you find some software bugs in the tool?
- Did the vendor make you aware of any bugs in the tool?

2.4 VENDOR SUPPORT ISSUES

2.4.1 Installation Assistance

A representative of the company that develops and/or distributes the rapid prototyping tool may be willing to visit your facility to assist in installation. If possible, it is helpful to have both a technical and a marketing representative in attendance. This will help you to begin using the tool as well as establish a basis for future support.

Evaluation Questions:

- Did the tool representatives visit your facility to assist in the installation procedure?
- What was the cost of these services?
- What type of assistance did they provide?
- Were they technically capable?

2.4.2 Evaluation Period

Vendor support can also include the opportunity to obtain a tool for a trial evaluation period. Evaluation periods allow the tool user to evaluate which tool will best suit a particular application's needs. They are typically limited to a 90 day duration, but can be extended in extenuating circumstances.

Evaluation Questions:

- Were you able to obtain the tool for a free evaluation period?
- How long was the evaluation period?
- Were there any difficulties in getting the tool to run?

2.4.3 Training

The tool vendor may be willing to provide a few days or a week of training, at either the vendor's or the user's facility. The location of the training can often depend on the number of people to be trained and on hardware platform availability.

The costs and benefits of a training program must be weighed in advance; this is a difficult task. There are several factors to consider. First, the vendor may agree to reduce the price of training per person if there are many trainees, or if more than one copy of the tool will be purchased. Secondly, the most effective courses usually require a trainer(s) with experience in both education and software engineering. Courses that divide the course time evenly between classroom lectures and "hands-on" assignments are helpful.

The answers to the above issues will assist the user in deciding whether to purchase a training period. As noted above, it is also useful to obtain the tool for a trial evaluation period to assess its difficulty before making a purchase decision.

Evaluation Questions:

- Did you receive any formal training from the tool vendor?
- Did the course mix class lectures with "hands-on" assignments?
- How long was the training course?
- Did it cover the features you needed for your application?
- Was the course tailored to your needs?
- How long did it take you to become proficient with the tool?

2.4.4 Telephone Assistance

The quality of telephone support for a tool can be critical for a smooth and successful prototype development. Indeed, assistance via telephone may be the most important aspect of vendor support. Evaluation, installation, and training support take place at the beginning of the process of using a tool; after they are completed, the telephone will probably be the sole source of vendor support. It is therefore important that technical representatives be available for telephone support. It is particularly helpful if a west-coast vendor has technical support representatives located on the east coast and vice-versa. Hours of "telephone tag" can cause long delays in prototype development. Finally, the telephone representatives should be fully versed in the history and technical details of the tool. When they do not immediately know the answer to a question, they should be willing to look into it and return the call later.

Evaluation Questions:

- Did the tool vendor supply you with prompt telephone support?
- Were there people specifically assigned to full-time technical support?
- Was it difficult to get someone available to help you?
- Was help available during your work hours?
- Did they provide an (800) phone number?
- Did they provide electronic mail access?
- Were they able to solve your problems?

2.5 SOFTWARE ISSUES

2.5.1 Standards Compatibility

To support the goals of portability and compatibility with other tools, rapid prototyping tools should adhere to the emerging industry standards such as distributed windows, dialogue style (e.g., Motif or Open Look), and communications protocol.

Evaluation Questions:

- Does the tool adhere to any current industry standards?

- If so, which ones?

2.5.2 Open Architecture

Many rapid prototyping tools on the market address a variety of applications. Their features often complement one another. For example, one tool may support development of a user interface, another may be a mapping display system, and a third may be a knowledge-based hypertext tool. It is helpful if these tools have open architectures so they can be easily integrated and used together. Tools with open architectures can accept the output from another tool as input, or provide output to another, enabling the user to put an array of tools to work on an interface problem.

Evaluation Questions:

- Did the tool have an open architecture?
- Did it accept output from another tool?
- If so, in what format?

Preference Questions:

- What types of tools would you want to use with it?

2.6 HARDWARE ISSUES

2.6.1 Portability

Portability can be an important factor for many applications. Not all prototypes are "throwaway" software projects. Many user interface prototypes are developed as a "proof of concept" for a system and then expanded into an actual system. And if the target hardware platform is not readily available in the design phase, portability can become an important issue. As hardware technology continues to evolve, system portability becomes even more of a challenge and a necessity.

Evaluation Questions:

- Does the tool enable you to develop a portable prototype?
- On what hardware platform can you build the prototype?
- To what hardware platforms can you port the prototype?
- Did the tool provide a run-time version?

2.6.2 Multiple Display Devices

As distributed systems become more common, it is important that rapid prototyping tools be able to support more than one display device at a time.

Evaluation Questions:

- Can the tool support multiple display devices?
- If so, how many and what types?

2.6.3 Multiple Input Devices

Rapid prototyping tools should be able to handle input from multiple devices at one time. In addition, they should be able to accept input from a variety of input devices, such as the keyboard, mouse, or track ball.

Evaluation Questions:

- Can the tool handle more than one input device?
- If so, what type?

SECTION THREE

TOOL REVIEWS

3.1. INTRODUCTION

This section contains the HFE/USI Group's evaluation results for the following USI rapid prototyping tools: VAPS, LUIS/SMS, SL-GMS, DataViews, TAE Plus, SET. Included for each tool is an overview description of the basic software components, prototype development steps, applications, hardware platforms and cost (site license costs are negotiable for all tools). The evaluation results for each tool then follow. The five criteria categories introduced in section 2 (capability, usability, vendor support, hardware, and software issues) are addressed for each tool. The criteria the tool directly supports, or which it can support with additional effort by the user interface developer, are discussed. The evaluation results are summarized in a set of "Tools-At-A-Glance" tables in the appendix.

3.2. VAPS

3.2.1 Overview

The VAPS Prototyping System is a high-end USI rapid prototyping tool. VAPS provides a flexible prototyping environment in which the user interface can be modified quickly and easily. VAPS is a product of Virtual Prototypes, Inc. of Montreal, Canada. Version 2.05 is the most recent version of VAPS and is discussed here. VAPS originally ran only on Silicon Graphics workstations, but it now runs on other platforms, including Sun and IBM (and soon HP) workstations. Its cost ranges from $15,000 to $41,000, depending on the configuration.

Although VAPS was originally developed to support avionics applications, it can also be used to support a range of user interface applications. It is particularly useful for prototyping hardware devices such as equipment and display consoles. Version 1.04 has been used successfully to prototype control panels and transceivers for a tactical system, command center displays, instrumentation panels, tactical radio panels, and aircraft cockpits. Version 2.05 has been enhanced to accommodate advanced graphics symbology for C3I and air traffic control.

The VAPS Prototyping System consists of four software subsystems that collectively provide all the capabilities needed to quickly implement a user interface. The subsystems include the object editor (also known as the graphics editor), integration editor, logic editor, and runtime environment.

3.2.2 Object Editor

The object editor provides an extensive set of drawing tools, including precision drawing aids and overlay templates, and can be used to create prototype displays, or in VAPS terms, "frames," and the graphic objects that comprise the frames. Graphic objects can be

precisely placed within the frames via a pointing device. The overlay templates permit a portion of a prototype frame to be updated or redrawn, decreasing response time.

To create the frames, the object editor supplies a variety of graphics primitives, attributes, and operations. The VAPS primitives include diagonal lines, horizontal and vertical lines, rectangles, circles, arcs, filled arcs, spline curves, filled splines, text, and filled polygons. There are four attributes: color, texture, line style, and font style. The operations that can be performed upon the objects include moving, rotating, scaling, duplicating, copying, replicating at an angle, deleting, corner rounding, and undoing the previous action. An object is created by selecting and grouping a set of primitives.

The object editor also allows the user to assign behavioral characteristics to graphic objects through values such as minimum and maximum, lines of motion, and center of rotation. There are three basic types of behavioral models for VAPS objects: display objects, input objects, and output objects. A display object is non-interactive. Input objects are objects with which the end user can directly interact. There are several different kinds of input objects.

- The button object is multi-state and can take on a different graphic representation based on its state. For example, a light can be in an "on" or "off" state. A button's state might represent one of several colors, changing, for example, among red, yellow and green based on three states.

- Switches are similar to buttons, except that each state of the switch is represented by a different part of the screen, while button states are represented by changes in color or other attributes at the same position on the screen.

- A rotor is an input object that generates discrete numerical values based on its rotational position.

- A knob is similar to a rotor, except that the knob generates continuous rather than discrete data values.

- A potentiometer generates data values based on its linear position.

- An event object signals the occurrence of an event. When selected at runtime, it sends a code back to an application system indicating that a specific event has occurred.

There are also several different output objects, which can be driven by the application system or directly by input objects.

- A light is a multi-state output object that can take on different graphic representations based on state values received as input.

- A dial rotates based on input values.

- A scale moves linearly .

- A tape moves behind a defined window so that only the part of the tape inside the bounds of the window is visible.
- A dial moves both rotationally and linearly.
- A barchart extends and contracts between a maximum and minimum value.

New objects added in version 2.05 include:

- A plan position indicator for radar and sonar.
- A graph object for charts and graphs.
- A menu object for pull-down and pop-up graphical menus.
- A locater for simulating 2D input devices.
- A CRT object for multiple nested virtual CRT displays.

Other characteristics of the object editor include an "infinite" drawing space, an overlay plane for references, and "precision points" for object placement. References are objects that appear on the screen to assist with object placement, but which are not saved as part of the frame.

3.2.3 Integration Editor

After creating the graphic objects, the next step is to connect the output objects to the input that will drive their behavior during runtime. This is accomplished within the integration editor. Output objects can be driven by input objects, by internal VAPS-supplied functions such as sine, triangular, and ramp functions, or by external simulation. The integration editor creates visual representations for the input and output plugs. Using these representations, output objects are "connected" to whatever input will drive them. Input objects are considered internal drivers, while VAPS-supplied functions and simulation values are external. Additionally, the integration editor provides a feature to combine various input data through mathematical or Boolean functions.

3.2.4 Logic Editor

The third step in user interface development is accomplished in the logic editor, which enables the user to specify and control the dialog with the prototype. It automatically generates code based on the frames, objects and connections established in the object editor and the integration editor. In VAPS, just as the objects themselves can have a number of states, the user interface as a whole is also based upon a finite-state machine model. This means that the user interface can transition between different states, for each of which a number of events are considered active or "executable." VAPS considers each new frame within an interface a unique state.

The code generated by the logic editor, Augmented Transition Network (ATN), is a high-level graphic interface language similar to the C language. In fact, C code can be embedded within ATN, and during compilation the ATN is translated to C code.

Using ATN, the developer defines those events which are active or inactive within each state. VAPS provides a programming library of over 100 routines that can be called inside the ATN to perform actions, such as highlighting objects, moving objects dynamically, changing object attributes, and reading text input fields. The only difficulty in using the logic editor is in planning and understanding the user interface dialog. A good approach, and one that is stressed in the VAPS training course, is always to draw a state diagram before beginning to modify the ATN.

3.2.5 Prototype Development Steps

The steps for building applications using VAPS include:

- Designing the frames that will make up the prototype
- Creating the frames and objects within the object editor
- Connecting dynamic objects to their appropriate drivers within the integration editor
- Designing a state machine model of user interface
- Generating and modifying the ATN within the logic editor

The display manager handles the runtime activities. After running the prototype, it may be necessary to return to the object editor, the integration editor, or the logic editor to make modifications. It is very easy to quickly modify any aspect of the interface until it functions as desired.

3.2.6 Evaluation

Capabilities

VAPS supports development of vector-drawn geographical maps, but a custom data format may be necessary. VAPS directly supports the World Database I and II and Defense Mapping Agency Digital Terrain Elevation Data (DMA DTED) formats. It also allows easy zooming and panning of maps and other graphic objects. Toggle overlays for mapping applications can be defined within VAPS by assigning visibility attributes to sections of the map.

VAPS supports both alphanumeric and picture labels on display objects. It is also possible to create formatted text input fields, but that type of interaction is not a major emphasis of VAPS. The same is true for creating objects that flash or reverse video: it can be done with VAPS, but it is not easy. For example, to create a flashing object, two states must be defined, the normal or "non-flash" appearance and the "flash" appearance. To toggle between the two states, the object must be connected to some type of input function, such as a sine wave. When

the value of the driving function is between 1 and 2, the object's graphical representation for state 1 is displayed; when the value is between 2 and 3, the state 2 representation is displayed.

VAPS supports a variety of input and output objects, but they are usually created from primitives within the object editor, unless the user first creates an object library. The palette of primitives that are directly supported includes circles, arcs, ovals, and rectangles. These primitives can be used to create the higher level objects such as potentiometers, dials, and knobs.

VAPS generates Augmented Transition Network (ATN) code that can be modified within the logic editor. The system also provides a library of many graphic interface routines to perform high-level functions. ATN is based on a state diagram that defines the flow of control for the program, and it must be modified by the developer to tailor it to the particular prototype. Straight C code can be embedded within it.

Usability

The object editor, integration editor, and logic editor of VAPS all allow the developer to build a prototype by directly manipulating graphic icons. For example, the graphic primitives available within the object editor are all arranged on a palette. The developer can simply select and drag the graphics to any location on the frame. VAPS also provides single-layer menu bars to handle such operations as grouping objects, placing objects in front or back of other objects, and saving a model. It also provides a tailorable interface within which the developer can specify which menus are available at any given time.

VAPS also has a unique and useful feature that allows precise placement of an object on a frame, called "precision points." These points exist on the corners of every object. As the mouse cursor moves closer to the precision points, there is a slight pull on the mouse towards them. Once the cursor is within an object's precision point, it can be placed directly on top of the point of another object, causing the objects to be placed precisely next to each other. This can be an extremely useful design feature. For example, without precise object placement, it can be difficult and time-consuming to perfectly align the objects when creating a button palette. Precision points make it easy.

The VAPS environment for user interface development is flexible. VAPS is a powerful tool and can support the development of complex prototypes. However, it does require a great deal of overhead, which can affect runtime performance. Therefore, when building a prototype with VAPS, it is important to keep optimization and performance in mind. VAPS is not overly difficult to learn to use. It provides a large number of confirmation checks. The documentation is helpful. In addition, VAPS has a automatic "save" feature that can be set at different time intervals so that a prototype under development can be periodically saved. Some limitations are presented by the system's somewhat closed architecture, in that it is limited to a particular format for graphic input files. However, VAPS is compatible with AutoCAD and PostScript files, and it provides device support for non-standard hardware in the runtime environment.

Vendor Support

Installing VAPS is a straightforward procedure that does not require additional assistance from the vendor. VAPS is available for a trial evaluation period. It also offers excellent training courses. The classes are conducted by both a professional educator and a software engineer. Each day of the week-long course is divided between lecture and hands-on exercises. It costs $2,500 per person plus traveling expenses. The courses are limited to a maximum of 6 students totaling $10,500. Virtual Prototypes, Inc.'s telephone assistance is helpful; however, all software bug fixes are saved for each new release. They will not send you a tape immediately with the fix, as some vendors do.

Software/Hardware

As it does not run under a standard windowing system, VAPS cannot be considered to be compatible with industry standards. Version 2.05 runs on the Silicon Graphics, IBM and Sun workstations. It is currently being modified to run on more hardware platforms. It also accepts data across an Ethernet network, TCP/IP, and 15553B from an external simulation or database, and accepts input from multiple display devices, including touchscreen, mouse, track ball, and keyboard.

3.2.7 Summary

Because its emphasis is on objects such as lights, buttons, knobs, and dials, VAPS is well-suited to modeling hardware devices. The new objects in version 2.05 make it possible to model complex display consoles. Another major strength is that VAPS enables a developer to quickly and easily build and modify a user interface. It is a high-end, powerful rapid prototyping tool. It can significantly reduce the development time of a prototype, in contrast to the time required by many other commercial rapid prototyping tools. However, code optimization should be kept in mind to minimize overhead and maximize response time. Another drawback is that VAPS is expensive. Version 1.04 costs from $50,000 to $100,000. The price of 2.05 was reduced to $15,000 to $41,000. Finally, because of its overhead requirements, it is only feasible to run VAPS on high performance workstations, such as the Silicon Graphics.

3.3 LUIS/SMS

3.3.1 Overview

LUIS/SMS consists of an integrated environment of two separate but related systems. The first is the Lockheed User Interface System (LUIS), which provides a deeply nested menuing system to support development of a user interface. The second is the Softcopy Map System (SMS), which supports development of geographical mapping systems. Although LUIS and SMS can both run as stand-alone systems or as an integrated package. LUIS provides the interface through which the user can interact with the mapping system. Moreover, without the advantages of SMS' special mapping capabilities, there are many user interface development systems that are easier to use than LUIS. As a package, however, LUIS/SMS provides a unique set of capabilities.

The version described in this paper are LUIS 6.33b and SMS 6.63. The newest version, LUIS II, is currently being distributed. It runs on the Digital Equipment Corporation (DEC), Hewlett Packard (HP)/Apollo, Silicon Graphics, Inc. (SGI), and Sun 4 workstations. The entire system costs approximately $50,000.

LUIS/SMS was designed by Lockheed to support military applications that involve geographical maps. It can support many different levels of map detail, and it provides mechanisms for ensuring optimal performance. The LUIS user interface tool will be discussed first, followed by a description of SMS capabilities. Finally, there will be a discussion of the integrated LUIS/SMS environment.

3.3.2 LUIS

LUIS is a user interface development system that provides a menu-based editor as well as a command language. Its unique features include easy integration of the user interface with an external application program, and user input logging, which can be useful for usability testing of a user interface. LUIS consists of three major components: a dialog editor, a graphics editor, and a parser. The dialog editor supports creation and modification of user interface displays or "panels" using menus and single-letter commands. It also automatically generates user interface language code of the graphic user interface in an "author file." LUIS objects consist of visible objects, including panels, buttons, and menus, and functional objects that supply logical functions, such as AND, NAND, and XOR evaluators.

Visible Objects

LUIS visible objects are interactive objects that can be seen by the end user. A complete list of visible objects includes panels, pop-ups, buttons (or, as they are called in LUIS, options), rectangles, text, text fields, and analogs. A panel is a distinct screen in the user interface. A pop-up is a window that is located within a panel. An option is a selectable button. A text field is an area in which the user may enter alphanumeric data. Finally, an analog is a measurement device, such as rulers, meters, and gages, to which the user can assign numeric values.

The available attributes differ for each type of object. Panel attributes include text color, background color, progress color, text font, cursor aid, cursor pattern, mouse input, keyboard input, and time and date. The attributes applied to the panel are also applied as default attributes to its "children," objects that reside on the panel. For example, if no text color has been specified for a text input field, the text will have the color defined for the panel. "Progress color" applies to analog indicators, such as the mercury in a thermometer. "Cursor aid" is a category of attributes that apply to objects as the cursor interacts with them. For example, if the color or pattern of a button should change when the mouse cursor is over it, the cursor aid would define those attributes. The cursor can be a small rectangle, a crosshair, or a pointing arrow. The time and date are also supplied directly to the panel as attributes.

The pop-up can contain a group of objects, or become a pop-up window displaying a scrollable text file. The unique pop-up attributes are shadowing, dynamic position, cursor home, pull down, scroll bar, and dynamically created options. Option attributes include selecting a box size and specifying the implications or actions to be executed upon selection or

deselection of the option. The shape, scale factor, and indicator minimum and maximum values can be specified for analogs. "Shadowing" refers to a shadow that appears around a pop-up window, adding a "3D" look. "Dynamic position" allows the pop-up's position to be dynamically determined during runtime, based on the position of the cursor. The "cursor home position" is where the cursor will appear when a panel first appears on the screen. Other notable object attributes include specifying the size of a rectangle, defining a text string as blinking text, and declaring a text field to be formatted or justified, with numeric or alphabetic input values.

Functional Objects

Functional objects are actually operations to be performed. Functional objects can specify logical relationships between a series of options such as AND, NAND, and XOR, or they can be states, implications, or actions. A "state" specifies a logical expression to be tested during interface execution. For example, if an expression is evaluated to be "TRUE", then certain actions called "implications" will be executed. An "implication" specifies a list of interface operations that are carried out in sequence after a designated event, such as evaluation of a state to be "TRUE" or selection of an option. Examples of implications are enable, disable, select, deselect, save, copy, move, clear and object, fire an action, set an analog, go to a panel, refresh the screen, and exit the interface. An action calls a programming routine to be executed.

The visible and functional objects are created and integrated into the user interface in the dialog editor. The LUIS dialog editor is analogous to the graphics editor in many other rapid prototyping systems. It is that part of the system in which most of the user interface development is accomplished. LUIS also has a graphics editor, but it is used for special cases. For example, the graphics editor supports creation and tailoring of unique analogs and custom-made option icons. Objects built in the graphics editor must be brought into the dialog editor to be incorporated into a prototype.

The dialog editor produces a textual description of a graphic user interface in an "author file". The LUIS "author file" is an ASCII file containing a collection of object definition statements. Having this intermediate author file provides a user interface language alternative to the menu-based system of LUIS for user interface development. The author file parser interprets the high-level statements in the author file, and creates the display list during development and at runtime. Parsing is required the first time an author file is created and when the interface is modified. Parse time is negligible for small interfaces, but may be considerable for interfaces consisting of two or more panels.

3.3.3 SMS

SMS is a portable software tool that supports rapid development and integration of maps into C3I applications and can accept map data from many sources, including the World DataBank and USGS. It provides a library of 900 routines, all of which handle various aspects of map display generation, including map database access, terrain shading, graphic utilities, and system management. Terrain shading refers to special effects, such as range shading and peak definition.

The maps are displayed as a series of overlays that can be toggled on and off. The overlays can contain separate portions of the map, such as country boundaries, altitude shading, peak definition, major rivers, major roads, major cities, and city labels. The graphic utilities include the low-level graphic drawing necessary to support the more complex maps as well as transformations such as rotation and scaling. System management routines handle features such as map legends, panning, and zooming.

SMS consists of a worldmap, display configuration, device description, units, symbol format, and symbol databases. Its worldmap database, which defines what is available to be displayed by SMS, contains all the available map data. The display configuration database defines how the data is to be displayed, specifying special features and attributes. The device description database contains definitions of colors available for use in display configuration; it defines the RGB (red, green, and blue) components of all the available colors. The symbol format database contains definitions of the types of symbols or icons to be displayed on the map. These databases reduce the complexity of creating a map application with SMS; the developer has only to modify them to tailor them for a particular application.

LUIS/SMS functions as an integrated environment. LUIS handles the user interface development, accepting input requests from the user during runtime and calling the appropriate SMS routine. SMS handles the mapping capabilities and is treated as any application system by LUIS. Other application programs can be easily bound with a LUIS interface. LUIS is bound to application through a special "process selection" file, a programming routine that must be created by the developer. It accepts a process name as user input and then calls the appropriate process routine.

3.3.4 Prototype Development Steps

The steps involved in building an application with LUIS/SMS, not necessarily in this order, are as follows: design the panels, build the user interface within the dialog editor, create special graphic objects in the graphics editor, parse author file, modify the appropriate databases, and generate the special process selection file to bind LUIS and the application.

3.3.5 Evaluation

Capabilities

LUIS/SMS is one of the few rapid prototyping tools that combines a user interface development system with a geographical mapping system. This makes it particularly useful for military applications, as they often require the use of map displays. LUIS/SMS allows maps to be easily manipulated, zoomed, panned and reset. It also provides toggle overlays, which, for example, allow the major rivers to be removed from a map without having to redraw the entire display. Toggle overlays are important to performance in drawing complex maps with terrain shading and elevation shading, in addition to roads, rivers and cities.

Alphanumeric and text labels are considered part of the LUIS display objects, so the entire object must be recreated to change the label. Special highlighting effects such as reverse video and flashing are easy to implement. The system can also do user input logging.

The logs are time-stamped and saved in a formatted ASCII file; but the logging applies only to LUIS objects, not to SMS objects.

Usability

The LUIS interface description file, or author file, includes implication objects that specify the actions to be taken according to the end user's input. For example, when the end user selects a button, the implication may be that a pop-up appears on the screen. Upon de-selection, the implication might be to cause the pop-up to disappear. The developer can define the implications through the LUIS menuing system or by entering them directly into the author file. In this way, LUIS/SMS supports both novice and expert users. However, although the menus are intended to support the novice user, they are deeply nested and can be confusing to use. They are sometimes five or six levels deep, and some have twenty or more elements.

Objects can be placed on the screen according to a grid. The placement can be fine-tuned within the author file. The author file is automatically generated by the dialog editor, and can be modified by the developer.

Custom-made objects must be created in the graphics editor. Because custom-made objects are not a major emphasis within LUIS/SMS, its graphics editor is rather limited and difficult to use. The toggle overlays, an important feature of SMS, avoid serious degradation of performance for applications in which complex map data is being manipulated. Nevertheless, it is best to run SMS on a high-end workstation, such as the Sun 4.

LUIS/SMS provides a flexible environment for development. It does, however, require training, and has a steep learning curve. The documentation is usable, although it would benefit from review by a human factors person to minimize some of the technical terminology.

Confirmation checks are only used when exiting the LUIS/SMS system. However, LUIS/SMS has file recovery methods. As each new author file is created, it is assigned a ".mod" extension. Thus, the next to last version of the author file would have the extension ".mod.mod" and so forth. In addition, a directory called "rcb," which contains binary files describing the user interface, is automatically generated .

Vendor Support

LUIS/SMS is complex to install, and the vendor does not regularly send a representative to assist. The environment and the paths must be carefully set up. The best bet is to have it installed by a professional system administrator. However, the newest version, soon to be release, LUIS II is easier to install and setup.

LUIS/SMS can be obtained for a trial evaluation period. The training courses offered by Lockheed are sometimes poorly structured and are not always taught by professional educators, but Lockheed is currently trying to improve the training system. The courses run for a week at $4600 plus traveling expenses. They are limited to a maximum of 6 students. The steep learning curve associated with LUIS/SMS complicates the training process.

Lockheed is not always accurate and timely with its telephone assistance. However, it is willing to fix bugs immediately and send the user a temporary tape with the fixed version until the next version release.

Software/Hardware

LUIS/SMS is not compatible with a standard window system, but it does use Extended GKS as its lower level graphics package. It is portable to other platforms, but not as portable as it might be if it were based on a windowing system such as X Windows. It can accept most World Database data files as input, but again, it restricts the format that it can accept.

3.3.6 Summary

The strengths of the integrated LUIS/SMS package are numerous. First, LUIS enables the developer to create the user interface without programming. The developer can create both the "look" and the "feel" of the interface independent of any coding. Through a complex menuing system, the developer can create, position, and modify panels and objects and specify the functionality behind them. The menus allow the developer to specify an implication to be executed when an option is selected or deselected. LUIS requires coding much later in the development process than do many other rapid prototyping tools. Coding is necessary to bind the application programs (such as SMS) to the LUIS interface.

Another major strength of LUIS/SMS is that it combines the capabilities of user interface design with map display generation. This is a unique combination, and one that is very useful for C3I applications. LUIS/SMS provides good text handling capabilities. It makes creating a text input region or a scroll bar in a text pop-up easy. Its input logging feature is very useful for performing usability testing on the interfaces developed with LUIS/SMS. And it runs on a wide range of hardware platforms.

There are tradeoffs, of course. For example, the same complexity of the menuing system that allows the developer to create the "look and feel" of the interface independent of programming also can cause the tool's interface to seem complicated and cumbersome. Also, although programming is not required quite as early in the development process as with many other tools, the integrated LUIS/SMS does, in fact, require a substantial amount of coding to create a mapping application. All the related databases must be modified, and a process selection file must be created. The coding within these files has rather strict syntax requirements. It includes over three-hundred high-level geographical mapping, display object manipulation and inter-process communication (between LUIS and external applications) programming routines. There is a steep learning curve associated with the tool.

3.4 SL-GMS

3.4.1 Overview

The Graphical Modeling System from the SL Corporation in Corte Madera, CA is a general purpose rapid prototyping tool. It supports easy generation of graphic objects and assignment of dynamics to them. This evaluation was based on version 3.0f1, however, the current version 4.0 will also be discussed. SL-GMS runs on Sun, Megatek, Silicon Graphics,

VAXstation, and HP 9000/300/Apollo workstations. Its cost is $12,000 per license for development license and $2,500 for just a runtime license.

SL-GMS was originally designed to support the development of process control applications for manufacturing environments. However, it has evolved into a general purpose rapid prototyping tool.

SL-GMS consists of two main components. First, it provides an easy-to-use interactive graphics editor called "SL-Draw." SL-Draw supports development of dynamic graphic displays and objects. A display along with the objects that reside on it is called a "model." The "look" of a prototype can be created with SL-Draw, independent of coding. The user interface can be stored in both an ASCII file and a binary code file. The editable file has ".g" as an extension; the non-editable file has ".m1" as an extension. For example, if a model were saved as "space_data", SL-Draw would automatically generate "space_data.g" and "space_data.m".

Second, SL-GMS provides a function library called "SL-GMF" of over 1000 routines. These routines can be embedded within a C, FORTRAN, Ada, or Pascal control program. ("Control program" is the term that we will use to refer to the program that "controls" the functionality of a prototype, such as the sequencing of displays and the handling of input events.) Unlike the ".g" file, the control program must be developed by the prototype developer.

3.4.2 SL-Draw Graphics Editor

SL-Draw supports a wide range of graphic primitives, including rectangles, curves, triangles, circles, ovals, arcs, markers, text and splines. The attributes that can be applied to these objects include line style, line width, line color, text color, object color, object pattern, font size, font style, and marker style.

SL-Draw also supports a large number of operations that can be performed on the graphic objects. An object can be filled with a pattern, unfilled, rotated, moved, mirrored, scaled, and deleted. The snap, add, and delete points can be used to modify an object created with splines. A point connecting the spline drawings can be added, deleted, or snapped to a new position. With the splines, the last point can also be cancelled by the "backup" operation. Any type of operation that was performed last can be cancelled with the "undo" operation. Limited functionality for the display objects can be defined within SL-Draw, but a prototype requires a control program to make it fully functional.

3.4.3 GISMOs

In version 3.0f1, the only high-level graphic object provided was the selectable button. Version 4.0, provides an extension of the original graphics editor objects. It supplies a set of ready-to-use Graphical Interactive Screen Management Objects (GISMOs). The most significant change in Version 4.0 is that it is compatible with the X Window System. X Windows, an emerging industry standard, allows an application to be distributed across a network, sharing displays among a heterogeneous set of workstations. The X Toolkit is

software that supports implementation of the user interface "look and feel." Also, a Motif "look and feel" can be added to any application.

The SL-GMS GISMOs are intended to supplement the X Toolkit. They can be used to control input events, such as mouse or keyboard input. For example, "DATSCREEN" is a GISMO that displays a new screen and activates a file that contains the screen's dynamics. Similarly, the "SCREEN" GISMO displays a new screen. "NEW POPUP" displays a pop-up menu or a dialog box. "RETURN" causes the interface to return to the previous screen. "QUIT" exits the application program. "CALL" invokes the specified application function. "FLASH" calls an application function with "flash" highlighting. "STAYON" calls an application function with highlighting in which an object stays in the "on" state, however that is defined. "RADIO" calls an application function and allows only one GISMO to be highlighted at a time. "TOGGLE" calls a function with a highlighting capability that turns a GISMO on or off, depending on its current state. There are also text-handling and dynamic-handling GISMOs, such as "SLIDER" and "FILL PERCENT." The library of GISMOS can be expanded to contain custom-made objects.

3.4.4 SL-GMF Function Library

SL-GMF, the function library, contains over 1000 programming functions. After the screens and objects are defined, the next step is to create a control program with embedded SL-GMF routines. The SL-GMF routines provide high-level functions, such as initializing SL-GMS, defining the initial model, redrawing the screen, changing the color of an object, turning off the visibility of a pop-up window, and navigating to another model. Functional "hooks" to the control program can be defined within SL-DRAW. For example, if a display has two buttons, one button can call the "pop-up-window" routine with parameter (1) and the other can call it with parameter (2). The code then checks the parameter passed by SL-Draw and pops up the appropriate window.

3.4.5 Prototype Development Steps

To create a fully functional prototype using SL-GMS, the developer must accomplish two main tasks: create the displays in SL-Draw, and generate the control program. Creating the displays includes building the objects from the primitives, placing them within a grid on the displays, and defining the "hooks" to the control program. Because all functionality is defined in the control program, coding must begin early in the development process. The process of developing the control program can often be simplified by using at least the beginnings of a sample control program. Many of the SL-GMS calls in the main program are similar for every application. GMSRUN is a new program offered by SL Corp. that allows a prototype to run without a control program.

3.4.6 Evaluation

Capabilities

While SL-GMS allows the developer to draw a map using the free-draw primitive in SL-Draw, it does not directly support development of vector-drawn maps. Maps can be imported from outside sources, although that is not a simple process. And while SL-GMS

objects can be zoomed, panned and reset in SL-DRAW, they cannot be manipulated similarly by the end user unless low-level zoom and pan code is written in the control program.

SL-GMS provides text-handling features such as text input fields; however, it is comparatively difficult to make these fields formatted and editable. Text can be placed anywhere on a SL-GMS screen. Text can also serve as labels on buttons, and the labels can be modified without recreating the entire object. Version 4.0 of SL-GMS is compatible with the X Window System; however, it is difficult to use the graphics editor in any size but full-screen, as the palette and the menu bars become too small to work with.

Usability

Within the SL-Draw graphics editor, graphic objects are created and their attributes defined through a set of graphic icons. The developer can directly manipulate these objects with a pointing device such as the mouse. Most of the icons have picture labels that represent primitives such as circle, rectangle, arc and button. Operations such as rotate, scale, and duplicate also have picture icons. However, files cannot be directly manipulated; that must be done through operating system commands.

SL-Draw also provides a set of single-layer menus along a menu bar. The menus provide functionality such as selecting and grouping objects, turning the object placement grid on and off, and saving the ".m" or ".g" description of a model. The objects are located on the screen by snapping them to a grid that can be set at various block sizes. The object location can also be "tweaked" by changing the screen position numbers in the control program. However, this often requires many calculations, and is is more difficult after an object is size-scaled. There is also an "XForm" function that can assist in moving objects. Objects can easily be placed in front or back of other objects. There are sixty-four available drawing colors in the graphics editor are limited to sixty-four.

SL-GMS supports both the novice and the expert developer. The novice can use the graphic icon palette in SL-Draw, while the expert can create objects and define their attributes in command language in the control program. Both, however, must be able to develop the control program.

SL-GMS provides many confirmation checks within SL-Draw. It supplies obstacles to erasing a file, moving from one model to another without saving, and exiting the system unintentionally. It also creates backup files when the developer saves a new version of a file. The system produces both an editable interface description file and a binary one. This ensures that the changes made to the editable file are not automatically incorporated into the prototype unless the developer issues commands to do so.

SL-GMS provides a flexible development process. The functional links between the SL-Draw objects and the control program routines can be created in any order. The system will not produce an error if a routine is specified in SL-Draw that is not defined in the control program until compilation. There can be a steep learning curve associated with using the SL-GMF routines effectively, mostly due to the software bugs. The routines do not always perform as stated in the documentation. For this reason, it is sometimes necessary to perform programming tricks with GMF calls to accomplish a simple task, such as making an object

appear and disappear. Moreover, the functionality of the SL-GMS GISMOs is rather basic, and similar features are available through the Motif Toolkit. The documentation will present a problem for both the novice and the expert user. It does not always address the details necessary to create an executing prototype. The documentation for version 4.0 has been improved.

SL-GMS does not require much overhead, and so can run on a Sun 3 with acceptable response time. However, this minimalization of overhead forces the user to program rather early in the development process, as do the limited graphic objects that are provided.

Vendor Support

Installation of SL-GMS is not difficult. SL Corporation will provide trial evaluation periods and training classes. The training courses cost $100 per hour plus $250 per student per day. They are limited to a maximum of 3 students for 4 days. The company's telephone assistance can be helpful, although the hours are limited, since all the technical support is located on the West Coast. To address this problem, SL Corp. has set up a distribution office in Washington D.C.

Software/Hardware

To adhere to industry standards, Version 4.0 of SL-GMS became compatible with the X Window System. The GISMOs are SL-GMS' own extension to the X Toolkit.

3.4.7 Summary

SL-GMS is relatively inexpensive compared to some of the higher-end tools, and yet it is a good general purpose tool. It frees the "look" of the prototype from all graphics coding. Screen development is easy and flexible. SL-GMS requires programming early in the development process. Development of the interface requires programming in a standardized language such as C or FORTRAN and also learning the SL-GMF (function library) calls.

The learning curve associated with SL-GMF can be steep, though it is not as difficult as those of some more powerful tools. SL-GMS also minimizes overhead and frees the developer from worry over performance. It is well-suited to interfaces that only require buttons and text input fields, and do not require complex geographical maps.

3.5 TAE PLUS

3.5.1 Overview

TAE Plus is a portable, X Window-based software package that supports the design and development of interactive graphic user interface application systems. It was developed by the NASA/Goddard Space Flight Center and is distributed by COSMIC, NASA's software distribution center located at the University of Georgia. Version 4.1 was evaluated and is discussed here. Version 5.1 was released in May 1991. TAE Plus is based on the X Window System, X11 Release 3 and allows the developer to build a user interface under X Windows

without requiring an understanding of the complexities of X. However, it allows the developer to access the X toolkit and X library directly if necessary. TAE Plus runs on Sun 3 and 4, DECstation, and HP/Apollo workstations. Version 4.1 cost is $500 for the general public and $250 for the government. Version 5.1 costs $750 and $350 respectively.

TAE Plus supplies a wide range of ready-made, high-level display objects such as pull-down menus, check boxes, radio buttons, and scroll bars. It is therefore very useful for user interfaces that require many different interaction styles.

TAE Plus consists of two main subsystems: the WorkBench and the Window Programming Tools (WPT). The WorkBench, the primary tool for user interface development, can be used to interactively create or edit the graphic portion of the interface, store the result in a file, and generate an application program. The developer can then use the WPT library of programming routines to add functionality to the interface. TAE also provides a graphics editor called "Idraw," but it is used only for creating and customizing special data-driven object types. All the interface development is done in the WorkBench.

3.5.2 WorkBench

The TAE Plus WorkBench is a development tool that facilitates interactive design of graphic user interfaces. TAE Plus interfaces consist of panels and interactive objects. An interface is made up of one or more independent parent panels. The interactive objects, or presentation items, reside on the panels: buttons, icons, check boxes, radio buttons, statics, text, pull-down, text lists, page edits, text displays, workspaces, and data-driven objects. All these presentation types are interactively created, modified and positioned on a parent panel using the TAE Plus WorkBench.

3.5.3 Presentation Types

Some of the TAE presentation type names are not self-explanatory. It is important to understand their functionality. Choosing the proper presentation type can make interface development go much more smoothly. For example, the radio buttons are mutually exclusive, while check boxes are not. Most of the TAE Plus presentation items are interactive. The exceptions to this are the static text item, which is primarily a label, and the "text display" item, which allows the user to scroll through large amounts of text without making any changes. The text display item is a scrollable rectangular area designed to display large amounts of text.

- The button is a selectable object, and upon selection is automatically highlighted by a "reverse video" effect.
- An icon is a small, selectable graphic object that serves the same purpose as a button, but unlike the button can be customized by the developer in the bitmap editor.
- A check box is a selectable box with a text label. It returns either a "no" or "yes," or a "0" or "1" to the application, depending on the data type.

- Radio buttons are a labeled set of mutually exclusive choices arranged in one or more columns. Clicking on any choice will cause any previously selected choice to be deselected.

- The static item is non-selectable text, primarily used for labeling and informational purposes. This text may be changed by the application program at runtime.

- A text item is a text input field.

- The pull-down is a pull-down menu.

- A text list is a scrolling list of mutually exclusive text choices.

- The page edit item is a simple multi-line text editor that allows the end user to enter multiple values for an item. The page edit would be used when a text input field provides too small an editable field.

- A workspace is a subwindow within a TAE Plus panel that can display and maintain X11 graphics. It is referenced by both a name in the WorkBench and an X11 window ID. The workspace behaves as an independent X window within a TAE Plus panel, and will provide the prototype with various types of X Window events, such as "Exposure", "EnterNotify", and "ButtonPress". The workspace can be very useful, as TAE Plus does not directly support any complex, specialized drawing, such as is needed to prepare geographical maps.

The objects in 4.1 are based on MIT's Athena widgets; and the objects in 5.1 are based on MOTIF, a window manager developed by the Open Software Foundation

3.5.4 Data-Driven Objects

A data-driven object is a TAE Plus object that displays the value of application data in real time. There are several types of data-driven objects: movers, rotators, stretchers, discretes, stripcharts, and dynamic text. They can be created by the interface developer using the TAE graphics editor Idraw, or they may be read in from an existing file stored in Idraw format. A data-driven object consists of a static background and a dynamic foreground object. The foreground object changes relative to the value of application data. For example:

- The dynamic foreground of a mover could be an arrow that moves horizontally or vertically along a static background that resembles a ruler. Its movement reflects a value received from the application program.

- The dynamic portion of the rotator rotates around a pivot point.

- The stretcher can expand or contract horizontally or vertically, similar to a thermometer.

- Discretes consist of several picture objects, each associated with a single value. Only one picture is displayed at a time. The discrete can be thought of as a series of drawings which, when displayed sequentially, can give the effect of animation.

- A stripchart displays a dynamic application value as a fluctuating line.

- A dynamic text item consists of a text string that changes with the values received from an application. For example, dynamic text could be used to prototype an alert message changing from "low" to "medium" to "high."

Only the icon and data-driven objects can be customized by the developer within the TAE WorkBench.

User interfaces developed in the WorkBench are saved in a file called a "resource" file. The TAE resource file is a binary file. In addition to designing the "look" of the user interface within the WorkBench, TAE allows for a small portion of the "feel" -- the connections between panels -- to be defined. For example, a button may be linked with the appearance and/or disappearance of display panels. However, to define the functionality of the interface beyond simple panel connections requires programming.

3.5.5 Code Generation

To support software development, the WorkBench also provides an automatic code generation capability. The generated code is based upon the user interface "look and feel" as specified in the resource file. TAE Plus can generate code in C, Ada, or TAE Command Language (TCL). TCL is an interpreted command language. Although it provides slower performance, it allows the prototype developer to "rehearse" functionality prior to the program being interpreted.

The code generated by the TAE Plus WorkBench is highly structured and modular. It also contains comments and print statements for debugging purposes. The generated code contains structures called "event handlers" that handle the functionality of the interface. Event handlers are automatically created for the connections made in the WorkBench. Most of the modification required by the developer to tailor the code involves inserting WPT statements within the event handlers. The code generator also supports initialization of the panels, and allocates memory for a new set of interactive objects to be located on the panel, reading the resource (".res") file to obtain the new set of objects, and then creating the user panel and its presentation items.

3.5.6 WPT Programming Library and X Windows

The WPT routines are callable routines used to display and control an application's user interface. They define and control TAE Plus panels and interaction objects. For example, WPT routines allow the attributes of panels and presentation items to be created, modified, displayed, and deleted during runtime. They also handle input from the end user. The X Library and Toolkit provides the building blocks used by WPT to define the panels and presentation items. WPT allows the developer to create an X-based interface without having to understand the complexities of X.

However, the developer is free to access the X Toolkit and XLib directly. The X Toolkit, which sits on top of XLib, is a library of routines constructed with XLib functions. XLib is a library of programming routines that contains more than 200 graphic and windowing functions. The TAE Plus presentation types represent a subset of the functionalities offered by the X Toolkit widgets.

3.5.7 Prototype Development Steps

To create an executing prototype using TAE Plus, the developer must take the following steps in the order listed. First, the panels and presentations items can be created in the WorkBench. Care must be taken to create the presentation items in the order in which they will appear, going from back to front. Next, the developer can create or modify any custom-made icons or data-driven objects. The sequencing of the panels can be specified in the WorkBench by selecting "connections" mode. The connections can also be "rehearsed" in the WorkBench. Finally, when the developer is confident that the interface is complete, the code can be generated. This is done by choosing the "generate" option. The code should then be modified by inserting WPT calls into the event handlers and adding any special routines that were built externally.

3.5.8 Evaluation

Capabilities

Although TAE Plus does not directly support map creation, it supports the importing of graphics such as geographical maps through the workspace object. Once imported, these maps can be placed within a TAE Plus panel and easily zoomed and panned. TAE Plus also supports labels, both as part of a selectable object and as static text. The static text labels can also be highlighted with special effects such as flashing. In addition, the text handling capabilities are quite good. TAE Plus offers many high-level display objects that provide various types of text, such as static text, text list, text display, and the page edit. Text input fields and system output fields can be easily formatted.

TAE Plus provides many ready-made, high-level display objects, referred to as "presentation items," which are very similar to those of MOTIF and Open Look. The buttons, icons, pull-down menus, radio buttons, checkboxes, and text input fields are quick and easy to include on any panel. The system also supplies dynamic objects such as rotators, stretchers, and stripcharts, called "data-driven" objects; many user interface design tools do not have such a complete set of interactive objects. However, only the icons and data-driven objects can be defined and customized by the developer in the WorkBench editors; the other items cannot be drawn. The data-driven objects do not directly support any type of complex animation, and cannot be connected to any internally generated function. For example, they would not be suitable to support a C3I display that requires a map with satellites moving randomly across the screen.

Most of the development is done in the WorkBench via dialog boxes. Idraw is extremely limited and difficult to use. It is not intended to be used for extensive development efforts. TAE Plus does not allow much of the "feel" of the interface to be defined without

programming. The only functionality that can be added through the WorkBench is the panel connections. However, many tools that do support definition of functionality independent of coding cost significantly more than TAE Plus.

To support easy development of a user interface, TAE Plus provides a code generation capability. The robust TAE Plus code generator produces programs that are fairly easy to understand and modify. The code is generated in a modular, highly structured format with many embedded comments. The code has structures called "event handlers" within which the developer must insert code or modify existing code extensively to define the functionality of the prototype. It can produce a control program in C, Ada, FORTRAN, or the TAE Plus Command Language (TCL).

Usability

There is no direct manipulation of graphic icons and limited menuing in the WorkBench. Rather, TAE Plus uses a set of dialog boxes that support creation of panels and presentation items, definition of their attributes, and sequencing of displays. The dialog boxes prompt the developer for each necessary piece of information. For example, the first dialog box prompts the developer to enter the resource file name. The next provides radio buttons that allow the user to create a new panel or a presentation item. Each of these causes another more detailed dialog box to appear.

Dialog boxes are often considered to be easier to use than deeply nested menuing systems. However, they do not allow for any direct manipulation. There are small pull-down menus attached to the main development panel that supply functions such as rehearse, generate, and save. Object placement is handled by snapping presentation items to a grid. The placement cannot be "tweaked" in the resource file.

TAE Plus is limited in flexibility. The developer must be careful about the order in which presentation items are created. For example, if a static text item is created before a button, the label will be invisible when placed on the button. The rendering of the label can be quickly accomplished by bringing up the label's dialog box and selecting the "OK" button. Similarly, the inability to modify objects other than the icons and data-driven objects is a deficiency. Many users like to tailor the objects to their application. For example, they may want the scroll bar on the other side of the window.

TAE Plus' object inheritance rules can be limiting. The visibility of an item cannot be toggled without also toggling its parent panel. For example, if the developer would like to include a flashing "Alert" button in an interface, he must also have the parent panel flash on and off. Also, once the source code is generated, it is difficult to modify the interface so as to add or to remove items. This lack of flexibility is a serious deficiency.

A user interface prototype cannot be built with TAE Plus without programming. Although the tool provides a code generation function, it requires a great deal of modification. Most of the coding involves inserting WPT calls in the event handlers." The only functionality that can be defined within the WorkBench is the sequencing from one panel to the next.

TAE Plus does not support the expert user as well as it supports the novice user. It can take a user about two weeks to a month to become proficient with the system. All prototypes must be developed through the WorkBench dialog panels. The resource file are compiled into binary at development time. TCL files are ASCII files, and are interpreted at runtime. Use of the WorkBench does not require additional training. The tool supplies a helpful tutorial on the development of panels and presentation items. The WPT library requires the most time. The TAE Plus documentation is useful, particularly if TCL is the development language. However, TCL can have a steep learning curve of its own.

TAE Plus also has a problem with response time on slower hardware platforms. For example, on a Sun 3, TAE Plus can take several seconds to draw a panel. Finally, if the resource file has not been saved, TAE Plus will provide a confirmation check before the system is exited.

Vendor Support

The TAE Plus technical support office provides particularly accessible and helpful telephone assistance. They also offer half-day training courses at no cost.

Software/Hardware

TAE Plus distinguishes itself by being based on the X Window system. Much of the user interface rapid prototyping literature states that compatibility with X Windows is a goal for the future. It allows an X Windows-based interface to be created without having to include calls to low-level X11 routines in the application program. However, TAE Plus supports inclusion of independently drawn and maintained X11 Release 3/4 graphics in an application created with Version 4.1 of TAE Plus. Therefore, it can run on any hardware platform that can run X Windows. This also allows a prototype to be distributed across multiple hardware devices. TAE Plus can also handle input from both a keyboard and a pointing device.

3.5.9 Summary

TAE Plus is a cost effective user interface development tool. It is particularly useful for developing user interfaces that require a wide variety of interactive display objects and have a fairly well-defined interface design. It does not provide a flexible development environment, but it does provide an easy interface to the X Window system. It also provides many prefabricated interactive objects.

Perhaps the most important strength of TAE Plus is its price. It offers a rather complete set of high-level display objects at a fraction of the cost of the other tools discussed in this paper. Other X-based user interface development tools include Builder Xcessory, XBUILD and UIMX.

3.6 DATAVIEWS

3.6.1 Overview

DataViews is a product of V.I. Corporation. It is a general purpose rapid prototyping tool that allows the developer to build a dynamic graphic user interface. DataViews runs on a wide range of hardware platforms, including Sun, HP/Apollo, Sony, Data General, IBM RS/6000, VAXstation, DECstation, and Silicon Graphics. Version 9.0 does not support the Sony platform. It costs about $17,000 per license. Although DataViews Version 7.0 was evaluated for this paper, Version 8.0 and, the latest version, 9.0 are also discussed.

DataViews was originally developed to support process control within a manufacturing environment, but it has evolved into a general purpose rapid prototyping tool. It consists of two major components: the graphics editor DV-Draw, and the programming library DV-Tools. DV-Draw is a menu-driven, direct manipulation graphics editor that allows displays to be created, animated, and executed, independent of coding. DV-Tools is an extensive library of routines that are callable from user-supplied "C" code. These routines support the dynamics and functionality of the interface.

3.6.2 DV-Draw

The DataViews graphics editor, DV-Draw, supplies both graphic primitives, such as lines, rectangles, polygons, arcs, circles, text, and vector text, and high-level display objects. The display objects consist of graphs, moving drawings, and input objects. Version 9.0 provides ellipses, and supports images stored in the Graphics Interchange Format (GIF). In DataViews terminology, the graphs are "dynamic graph displays." There are approximately sixty-nine, in 9.0, graph types within DataViews, including multiple types of bar, line, point, vector, and radial graphs. The graphs are complex drawings that can be animated within a display.

Through a menuing system, various behavioral characteristics can be assigned to the graphs. The behaviors usually take the form of color or shape changes that can be based upon an input value from any type of data source, such as a data file, an external process, a mathematical function, a variable, or a constant. For example, the bars within a bar graph can be defined to incrementally elongate based upon a step function. Dynamic attributes, such as color, movement, scaling, visibility and text display, can be assigned to all DataViews objects.

The graph object consists of the graphic representation, its associated characteristics, such as axis tick marks, enclosing rectangles, and the data source. The moving drawing is really a type of dynamic graph display. It can be an object or a group of objects that move across a display. Moving drawings allow an object to move with real-time positioning. For example, a moving drawing can move as the user inputs values. Input objects are objects which allow the user to interact with a graph, such as a knob, dial, or pull-down menu. New objects, in 9.0, include the menu, text entry, slider, checklist, and radio, in both OpenLook and Motif styles.

The attributes that can be applied to DV-Draw objects include position, size, style, and dynamics. The operations which can be performed within DV-Draw include duplicate, resize,

move, fill, refresh, group, and zoom. Scale, mirror, rotate and reorder are the new 9.0 operations. DataViews also provides a "preview" capability, so that the developer can rehearse the display dynamics without exiting DV-Draw. The preview capability can be used to verify that external data being used to drive the objects is being transferred correctly. Version 9.0 also provides a "prototype" capability" that allows a prototype to be run under DV-Draw or under UNIX or VMS. Developing a prototype with DataViews does not always require programming. The developer can specify "rules" within DV-Draw to define the functionality of display objects.

3.6.3 DV-Tools

DV-Tools is an extensive library of 1000 programming routines that handle general utility functions, such as loading, displaying, and updating displays. There are routines to manipulate objects, to handle events, and to provide a device-independent interface for display devices. DataViews also provides the source code for many sample "C" routines that handle functions common to most prototypes. For example, there is a routine that can be used to handle display sequencing within a prototype. These routines can serve as examples, or can be directly incorporated into a program.

3.6.4 Prototype Development Steps

To create a prototype with DataViews, the "look and feel" of the display objects should be defined in DV-Draw. The displays and the objects can be created via direct manipulation of the palette options. Using the menuing system, the developer can tie the objects to input data sources. The dynamics can then be tested within DV-Draw through the preview function. Finally, a "C" program with embedded calls to DV-Tools can be created to control the remaining functionality of the prototype.

3.6.5 Evaluation

Capabilities

DataViews 7.0 and 8.0 do not directly support development of geographical maps, but map can be imported in a CAD format called "DXF." These objects can be easily zoomed or panned. Although DataViews does not directly support toggle overlays, it does provide "plane masking," in which bit planes can be grouped together and defined as visible or invisible. Labels are handled conveniently; object labels can be modified without having to recreate the object. Similarly, DataViews provides text handling capabilities, such as text input fields.

DataViews' high-level display objects consist of various types of dynamic graph displays, which can be animated based on user input, external data files, or mathematical functions. It also supplies a specialized type of graph called moving drawings that can handle real-time positioning, rather than following a predefined path. The animation of moving drawing graphs can be controlled via an input data source: thus animation can be created independent of programming. However, if a number of dynamic objects must be displayed at one time, it is best, in terms of performance, to develop a program to control the X and Y positioning of the objects. DataViews also provides scroll bars and pull-down menus as high-level display objects.

DataViews versions 6.0 through 9.0 can run under the X Window system. It is compatible with the following window managers: Motif, OpenWindows, SunView and DECwindows. A prototype built with DataViews can be distributed over a network. DataViews does not provide a code generation facility. It does provide many sample programs that handle common functions.

Usability

DataViews provides both direct manipulation of a palette of objects and operations and a menuing system. It also offers an excellent color palette. The menus are not shallow; they often consist of four levels, with many options at each level. The menus have become more complex as the capabilities provided by DataViews have increased with each new version. However, through a combination of direct manipulation and menuing systems, the developer can design an interface and define some of its functionality.

The DataViews development environment is flexible. Objects, attributes, behavioral characteristics, and programming can be done in any order. Object placement is handled using a grid whose size can be set to different widths. Development without coding is possible with DataViews as long as the "look and feel" of the prototype is kept simple. The preview capability is convenient to test the prototype.

The learning curve associated with DV-Draw can be substantial due to the nested menuing and the wide range of capabilities available. The curve steepens still more if the prototype requires programming with DV-Tools. Many sample programs are supplied to ease the burden of programming. However, understanding and modifying these programs requires a certain amount of expertise. It can take two weeks to a month to become familiar with DV-Draw and DV-Tools. The documentation covers all the capabilities provided by DataViews, but many subjects are treated superficially. There are also a computer-aided tutorial, DV-Tutor. DataViews provides confirmation checks whenever there is a possibility of losing work.

Vendor Support

Installing DataViews did not present any problems. It can be obtained for a trial evaluation period. V. I. Corporation offers 3 day training courses at $1000 per person. Telephone support was found to be helpful, and somewhat compensated for the poor DV-Tools documentation.

Software/Hardware

DataViews is compatible with X Windows. It is portable to any machine that is running X and can be distributed across a network. It can accept input from the keyboard, mouse and touchscreen It can accept data in any format to drive the dynamic graph displays . In fact, the format can be specified by the developer. The data format is limited, however; geographical map data must be in a particular CAD format.

3.6.6 Summary

DataViews is a general purpose rapid prototyping tool that is relatively affordable. It is particularly useful for prototypes that require a great deal of dynamic graphics. DataViews provides a usable graphics editor that allows both the "look and feel" of the prototype to be defined without programming. It also provides a preview function that allows the developer to test the functionality of the prototype within the graphics editor. Its dynamic objects can be driven by any type of data source, and they can be defined to move across the screen with real-time positioning.

3.7 SET

3.7.1 Overview

The Software Engineering Toolkit (SET) is a software tool designed to prototype, develop, and maintain portable user interfaces. Version 3.4 of SET was evaluated. There have been significant improvements incorporated into the most recent version, 3.8. It is a product of CASET/PA Corporation of San Juan Capistrano, CA. SET is available on various platforms including Sun, HP/Apollo, DEC VAX, DECstation and VAXstation, Silicon Graphics and the PRIME 50 Series.

SET consists of a suite of programming libraries: windowSET, graphicSET, inSET, dataSET, and onSET. The cost of all five modules, including five days of on-site training and installation, is $7,500. The components can be purchased separately, or in various combinations. The price of the individual modules is under $2,000 each.

3.7.2 Draw Graphics Editor

The newest version of SET includes a number of demo programs that can be used as sample programs. One demo is called "draw.demo" which can function as a graphics editor. However, DRAW is an unsupported portion of the SET product. Having been created with the SET programming libraries, it was intended to be used as a sample application program. However, the SET technical support representatives suggest using it as a graphics editor. The fact that it is unsupported means that the SET technical support team is not responsible for fixing or recommending alternative methods to work around software bugs. Graphic displays generated in Draw are stored in files called "metafiles." The contents of each metafile is drawn as a "graphics frame" at the appropriate time during execution of the prototype. A "frame" is SET's term for a display.

There are, however, several problems with using Draw as a development tool. First, it contains quite a number of software bugs. Second, as an unsupported part of the product, it is not covered by the telephone support or maintenance agreements. Third, its use in building graphic frames will considerably slow down the performance of a prototype. Moreover, it is difficult to use. There are two alternatives to building graphic frames with Draw. One is to use the facility "Simview" which was released with version 3.6. Simview allows the developer to interactively design windows, menus, buttons, and so on. The other is to use the graphicSET programming library. Using graphicSET will avoid all the previously mentioned limitations. Our discussion of SET will therefore concentrate on the programming libraries.

It is necessary to define SET terminology before discussing the library routines. SET frames, or displays, are generated on what SET calls "drawing pads." A SET pad is an infinite drawing space that can contain both text and graphics. Since a pad is infinite, it is desirable to view only a portion of it at a time; a SET "window" allows a section of a pad, specified by the developer, to be viewed on the screen. A window is mapped onto a pad by specifying a "frame." A SET frame establishes the boundaries of a SET window.

3.7.3 Programming Libraries

Building a prototype with SET version 3.4 requires a great deal of programming. Version 3.8 allows an interface to be designed without programming. The five components that comprise the SET tool are windowSET, graphicSET, inSET, dataSET, and onSET. Each module is a library of special purpose routines. The five modules support windowing, graphics, interactivity, data structuring, and portability, respectively. WindowSET is a portable window manager that supports windowing capabilities on a wide range of text and graphics terminals. Version 3.6 and version 3.8 are also compatible with the X Window System;.3.8 is compatible with Motif. GraphicSET is a graphics package that provides both 2D and 3D drawing capabilities. InSET is the interaction handler that provides techniques for controlling the functionality of the user interface. DataSET is a data-structuring tool. And finally, onSET insulates the application program from the operating system to maintain the portability of the software. Together, these five modules provide the functionality to handle every aspect of prototype development.

WindowSET

WindowSET is a screen management system. WindowSET routines make a prototype compatible with the windowing systems on Sun and DEC workstations, SunViews and DECWindows. Version 3.4 is not compatible with the X Window System. Compatibility with X is a goal for future versions.

WindowSET functions on two separate levels. It has the capability to provide both the end user and the developer with control over a prototype. With the windowSET routine library, the developer can create and control windows of various pre-defined types. With the screen formatting facilities, the end user can modify the screen presentation while an application is running. Version 3.8 allows the screen presentation to be modified interactively.

With the windowSET library, the developer can create several different types of windows. The various window types include transcript, read, message, edit, VDU, and graphics windows. Transcript windows provide a work area for the interaction between the user and the application program. For example, they can contain the application output as well as an echo of the user input from the command line. Read windows allow the user to list and peruse (read-only) the contents of a file. Message windows contain help messages and additional information supplied by the application program. Edit windows are similar to read windows, but have both read and write capabilities. VDUs (visual display units) are windows specifically formatted to contain text input regions, with attributes such as highlighting and color. Graphics windows are used for the display of graphic entities.

The windowSET screen formatting facility supplies the end user with such capabilities as window selection, frame control, pad control, menu control, and text editing, or, if it is available, windowSET relies on the X Windowing System. As with most windowing systems, a window is selected simply by placing the cursor within the region of the window. Frame control includes resizing, moving, copying, refreshing, deleting, popping, and pushing the window specified by the frame boundaries. Providing that a particular window has been defined as optional by the developer, the user has the ability to delete that window entirely from the screen. The user can scroll the pad in any direction to bring a different portion within the window frame. The user can also zoom in and out on the pad.

GraphicSET

GraphicSET is a graphics package that supplies the prototype developer with a library of routines supporting display generation. It provides all the capabilities necessary to perform lower level graphics, such as drawing lines and circles. GraphicSET has not changed from versions 3.4 to 3.8, except that the documentation has been improved. GraphicSET supports creation of the following primitive objects: lines, circles, circular arcs, ellipses, conic arcs, markers, and text. All the primitives can be combined to compose high-level graphic objects or pictures. There are actually two types of text: graphic and annotative. Graphic text is transformed with its parent object as it is rotated, scaled, and translated. Annotative text is designed to be used as labels and titles, rather than as part of a graphic picture, so it is not transformed with a graphic object. The attributes consist mainly of line styles, font styles, and fill patterns.

To accommodate some of the more complicated graphics, SET has a special file, the hierarchical display file (HDF), that stores primitives (such as lines and circles) before being displayed on the screen. The HDF allows a complicated drawing to be broken down into its respective components and manipulated individually. It improves performance by eliminating the need to redraw the complete picture when only a portion of it is modified. The HDF's basic building block is the segment, consisting of a number of graphics primitives grouped together. The primitives can be a complete drawing or parts of a drawing. Each segment owns its own primitives. A segment may also own other segments. There are routines that open and close the segments. Once the prototype displays have been developed, it is necessary to provide a mechanism for user interaction.

InSET

InSET, the interaction manager of SET, is the module that generates the control program. The generated control program consists of skeleton code defining the events that may take place during the running of the prototype. It can be generated in C or FORTRAN.

The control program requires extensive modification by the developer. The logical flow of these events is defined in a separate file, called a "network" file, which must be created by the developer. It is structured as a textual description of a state machine diagram. Therefore, it is easiest for the developer to create a state machine diagram defining the flow of control of the prototype, and then convert this diagram into a network file. InSET compiles the network file into executable code.

DataSET

DataSET is a data structuring system that provides a suite of routines to enable the developer to store and retrieve data. It allows the user to construct and search a data structure without requiring any knowledge of how it is stored internally. DataSET provides data handling capabilities such as direct pointers, lists, and tree searching, and handles "schemas," which are compressed descriptions of the data structures.

In addition to the dataSET library, the SET developer is provided with a database development tool called IdS (Interactive dataSET). IdS combines the capabilities of dataSET and inSET to create an interactive environment for both the novice and the expert database developer. IdS prompts the inexperienced user at each step, while an experienced user who enters full commands is not bothered with the prompts. IdS contains the syntax to define the database schemas and to create new databases from those schemas. It also assists the developer in opening and manipulating existing databases. The SET data structuring facilities can be used alone, or in conjunction with a full database management system.

OnSET

The SET portability manager, onSET, provides a wide range of operating system independent routines. Features such as message handling, file handling, input/output, and character manipulation are handled by onSET, which organizes the software so that all the routines within a prototype that interact with the operating system are grouped together in a library of files. Creation of such a set of system independent functions is what ensures the portability of a prototype. The onSET software library is available on HP/Apollo, VAX, DECstation, Silicon Graphics, and the Sun 3/4 workstation.

3.7.4 Prototype Development Steps

Creating a prototype with SET requires several steps. First, the developer must design the frames that make up the interface. If the frames will be programmed rather than built in the Draw demo, the developer should generate the generic application program at this time. The code should then be modified with calls to the graphicSET library to create the interface. Once the "look" of the interface has been completed, the developer should begin to define the functionality. The best approach is to define the flow of control of the application within a state diagram, which is then easily convertible into the network description file format. InSET can integrate the network description into the application program. Depending on the needs of the application, routines from the other SET modules can be incorporated into the control program.

3.7.5 Evaluation

Capabilities

SET supports zooming and panning of graphic entities, although it does not support development of geographical maps. A new module, which became available in September 1991, will provide raster and vector graphics capabilities. CASET believes this will make the tool more useful for developers working with geographical applications. Within windowSET, SET provides a specialized type of window called a "visual display unit" that functions as a text

input field. This window can be highlighted with color and special effects such as reverse video. SET also supplies two distinct types of text labels: graphic and annotative. Graphic text can be used as part of a graphic picture; annotative text can be used as labels and titles. The main difference between them is that graphic text is transformed as its parent object is rotated, scaled, or translated; annotative text remains unaffected.

Version 3.4 of SET did not provide any high-level objects. For example, it did not directly support selectable buttons, which had to be created as one-item menus; each button requires approximately six lines of code. Implementing special features for the buttons also presented a problem. For example, the buttons do reverse video as the cursor moves over them, but they repeatedly blink while attempting to return to their original appearance. Many of these problems have been eliminated by 3.8's compatibility with X and Motif. In addition, the blinking problem has finally been fixed.

SET supports windowing systems, such as DEC Windows, SunViews, Display Manager X Window System, and Motif. It also provides a code generation facility. The generated code is a skeleton of the control program, and requires a great deal of modification. The code can be generated in C or FORTRAN.

Usability

Direct manipulation of graphic icons is available within the Draw demo or Simview. Simview allows the developer to interactively design displays. Simview is available in the new version, however, it has not yet been evaluated. However, use of the Draw demo is not recommended. Besides the fact that it is unsupported, during evaluation it produced an unacceptable response time. For example, it was found to take up to twenty seconds to sequence from one display to the next. With the programming libraries, this delay can be decreased to approximately five seconds, but that is still poor performance. However, CASET reports that they have evaluated the performance of the latest version, and it is good even when running under X.

With the older versions, if the Draw demo was not used, all development had to be accomplished through programming. For example, the display objects are placed on a display by specifying the exact screen X and Y coordinates as parameters to a routine.

There is a big learning curve for using the SET programming libraries. The learning process was also hindered by the poor quality of the SET documentation. The documentation of many of the library routines was ambiguous and incomplete. Many of the details necessary to build a prototype were not mentioned in the documentation. The graphic SET documentation, in particular, and much of the other manuals were improved for 3.8. They were changed from a reference format to a user's guide format.

IdS, the database development tool, combines the capabilities of dataSET and inSET to create an interactive environment for both the novice and the expert database developer. IdS prompts the inexperienced user at each step, while an experienced user who enters full commands is not bothered with the prompts. However, this is a small portion of the SET product. The other components of SET all require a high level of expertise for development.

Vendor Support

SET can be obtained for a trial evaluation period. The vendor may also agree to send a representative to assist in installation. Because of the weak documentation, training is crucial. Although the documentation has been improved, training is still recommended for the novice SET user. Training courses are offered for 3 days at $1000 per day plus traveling expenses for a maximum of 15 students. The CASET telephone assistance was exceptionally helpful.

Software/Hardware

The onSET routines can be used to port an application among the following hardware platforms: HP/Apollo, VAX, DECstation, HP 9000/Apollo, Sun 3/41 Sparc, and the Silicon Graphics workstation series. DataSET allows external data to be accessed by a SET prototype. It can drive more than one display device at a time, and can accept input through the keyboard and the mouse.

3.7.6 Summary

SET is version 3.4 a useful tool for expert developers who would rather develop a prototype through high-level programming with the routine libraries than through use of a menuing system or direct manipulation. Unlike many of the other tools, SET has undergone an extensive set of modifications during the last two releases. An interactive graphics editor was added; the documentation was improved, and the tool was made X/Motif compatible. Use of the libraries can be a significant improvement over coding from low-level graphics packages, such as CORE or GKS. However, the SET programming libraries can present a significant learning curve even for expert programmers, simply because of the poor documentation.

SECTION FOUR

TOOL SELECTION

As the evaluations included in this paper illustrate, there is no one "best" USI rapid prototyping tool. Rather, selection of a tool must be based upon a system's particular needs. Each of the tools evaluated has its own strengths and weaknesses, and these should be fitted to the system requirements.

For example, if a prototype requires high-level display objects such as scrolling text windows and pull-down menus superimposed on a set of complex geographical map data, then LUIS/SMS would be chosen. If, on the other hand, the prototype is to be a simulation of an Air Force display console, complete with dials and buttons, VAPS would be the appropriate tool. DataViews and SL-GMS might be considered for development of a more general purpose prototype, such as different applications of computer information systems. TAE Plus, on the other hand, may be chosen for a prototype which requires many high-level display objects such as radio buttons, check boxes, and scroll bars, but which has a limited budget to spend on tools. SET is particularly useful for developers who wish to do low-level graphics programming and have complete control of their development details.

In addition to considering the functional requirements of a prototype, it is important to consider the skills and training of the developers. For example, developers who do not have a great deal of experience with software development may choose to use DataViews over SL-GMS, as DataViews requires less programming.

To standardize the evaluation procedure and to provide a basis for comparing the strengths and weaknesses of the tools, we evaluated them against a list of criteria that cover a wide range of capability, usability, vendor support, software and hardware issues. This list can apply to many types of applications, but it focuses particularly on command and control user interface requirements.

The "Tools-At-A-Glance" tables help the prospective user determine the "best" tool for an application. They show how each of the tools rated against the criteria. The user can simply select a subset of criteria that are critical to a particular application, and pick the tool that is rated highest against them. However, it may be beneficial to assign weighted factors to the criteria subset according to importance. For example, if mapping capabilities are the most important factor, assign it a .20. Direct manipulation is of medium importance, assign it a .15. Next, assign numerical values, such as three, two, and one, to the three ratings: check plus, check, and check minus, respectively. Now simply multiply the numerical rating score by each weight factor and add up the total for each tool. The tool with the highest total score can then be considered best suited for the application. A sample decision-making chart is shown in figure 1.

The criteria list given in this paper can also be used as a basis for other evaluations. If the reader wishes to conduct an evaluation on a tool not included in this paper, he or she can use the evaluation criteria as a starting point. First a list of pertinent criteria should be

IL2690-1

Sample Criterion	Weight	VAPS	LUIS/SMS	SL–GMS	TAE Plus	Data Views	SET
Geographical Mapping	.20	2 .4	3 .6	2 .4	2 .4	1 .2	1 .2
Labels	.20	2 .4	3 .6	3 .6	3 .6	3 .6	2 .4
Text Handling	.20	2 .4	2 .4	2 .4	3 .6	2 .4	2 .4
Direct Manipulation	.15	2 .3	2 .3	2 .3	1 .15	3 .45	2 .3
Usable Documentation	.15	2 .3	2 .3	1 .15	2 .3	1 .15	2 .3
Training	.10	3 .3	2 .2	2 .2	2 .2	2 .2	2 .2
	Total	2.1	2.4	2.05	2.25	2.0	1.8

Figure 1. Sample Decision Matrix Chart

compiled, either as a subset or an extension of the one in section 2. Next, a model benchmark prototype, incorporating all the pertinent criteria should be designed. The tool should then be used to implement the benchmark application. The degree to which the tool supports each of the criteria can be rated on the basis of this "hands-on" experience with the tool. To assist in this process, the author will provide an on-line version of the tools-at-a-glance tables will be provided upon request.

In conclusion, USI rapid prototyping tools can offer a critical advantage to the developer of a high-quality user interface. Prototyping brings a design to life, enabling the user to "see and feel" the system specifications in action. Prototyping tools offer the power and flexibility to experiment with design options before the development stage. Because these tools are often expensive, MITRE/ESD projects may want to delay purchasing one until they are confident it suits their needs. This is where the HFE/USI Specialty Group can make a difference. Using our in-house library of USI prototyping tools, we offer the prospective user demonstrations and assistance in tool use, as well as new tool evaluations on request. And as new USI rapid prototyping tools emerge on the market, we will continue to evaluate them, and will update this paper with the results.

APPENDIX

"TOOLS-AT-A-GLANCE" TABLES

The "Tools-At-A-Glance" tables summarize the user interface rapid prototyping tool evaluations. Each tool is rated against the criterion on a three point scale: check plus, check, and check minus. "Check plus" means that the tool directly supports the criterion. A "check" tells the reader that the tool does support the particular feature, but it will require extra effort to implement. And finally, "check minus" means that the tool does not support the criterion. For criteria such as compatibility with multiple input devices or hardware platforms, a "check plus" means that the tool can support more than two, a "check" means that it supports two, and a "check minus" means that it supports only one. These tables can be used to determine the tool "best" adapted to an application.

IL2691

Table 1. VAPS Capability Ratings

Criteria	Rating		
	✔+	✔	✔–
Capability Issues			
Geographical Mapping		✔	
Graphics Manipulation	✔+		
Toggle Overlaps		✔	
Labels		✔	
Text Handling		✔	
High-Level Display Objects		✔	
Windowing			✔–
Code Generation	✔+		
User Logging			✔–

IL2692-1

Table 2. VAPS Usability Ratings

Criteria	Rating		
	✔+	✔	✔–
Usability Issues			
Graphic Icons	✔+		
Direct Manipulation	✔+		
Shallow Menuing	✔+		
Object Placement	✔+		
Environment Flexibility		✔	
Screen Overview	✔+		
Response Time			✔–
Development Without Coding		✔	
Novice Versus Expert User Support			✔–
Learning Curve		✔	
Usable Documentation		✔	
Confirmation Checks	✔+		
File Recovery		✔	

IL2693

Table 3. VAPS Vendor Support Ratings

Criteria	Rating		
	✔+	✔	✔–
Vendor Support Issues			
Installation Assistance		✔	
Evaluation Period		✔	
Training	✔+		
Telephone Assistance	✔+		

IL2694

Table 4. VAPS Software Ratings

Criteria	Rating		
	✔+	✔	✔–
Software Issues			✔–
Standards Compatibility			✔–
Open Architecture			✔–

IL2695

Table 5. VAPS Hardware Ratings

Criteria	Rating		
	✔+	✔	✔–
Hardware Issues			
Portability		✔	
Multiple Display Devices			✔–
Multiple Input Devices	✔+		

IL2696-1

Table 6. LUIS/SMS Capability Ratings

Criteria	Rating		
	✔+	✔	✔–
Capability Issues			
Geographical Mapping	✔+		
Graphics Manipulation	✔+		
Toggle Overlaps	✔+		
Labels	✔+		
Text Handling		✔	
High-Level Display Objects	✔+		
Windowing			✔–
Code Generation		✔	
User Logging	✔+		

IL2697-1

Table 7. LUIS/SMS Usability Ratings

Criteria	Rating		
	✔+	✔	✔–
Usability Issues			
Graphic Icons	✔+		
Direct Manipulation	✔+		
Shallow Menuing			✔–
Object Placement		✔	
Environment Flexibility		✔	
Screen Overview			✔–
Response Time		✔	
Development Without Coding			✔–
Novice Versus Expert User Support	✔+		
Learning Curve		✔	
Usable Documentation		✔	
Confirmation Checks		✔	
File Recovery	✔+		

IL2698-1

Table 8. LUIS/SMS Vendor Support Ratings

Criteria	Rating		
	✔+	✔	✔–
Vendor Support Issues			
Installation Assistance			✔–
Evaluation Period		✔	
Training			✔–
Telephone Assistance		✔	

IL2699

Table 9. LUIS/SMS Software Ratings

Criteria	Rating		
	✔+	✔	✔–
Software Issues			
Standards Compatibility		✔	
Open Architecture		✔	

IL2700

Table 10. LUIS/SMS Hardware Ratings

Criteria	Rating		
	✔+	✔	✔–
Hardware Issues			
Portability		✔	
Multiple Display Devices			✔–
Multiple Input Devices		✔	

IL2701-1

Table 11. SL-GMS Capability Ratings

Criteria	Rating		
	✔+	✔	✔–
Capability Issues			
Geographical Mapping		✔	
Graphics Manipulation		✔	
Toggle Overlaps			✔–
Labels	✔+		
Text Handling		✔	
High-Level Display Objects		✔	
Windowing			✔–
Code Generation			✔–
User Logging			✔–

IL2702-1

Table 12. SL-GMS Usability Ratings

Criteria	Rating		
	✔+	✔	✔–
Usability Issues			
Graphic Icons	✔+		
Direct Manipulation		✔	
Shallow Menuing	✔+		
Object Placement		✔	
Environment Flexibility		✔	
Screen Overview			✔–
Response Time	✔+		
Development Without Coding			✔–
Novice Versus Expert User Support		✔	
Learning Curve		✔	
Usable Documentation		✔	
Confirmation Checks	✔+		
File Recovery			✔–

IL2703

Table 13. SL-GMS Vendor Support Ratings

Criteria	Rating		
	✔+	✔	✔–
Vendor Support Issues			
Installation Assistance		✔	
Evaluation Period		✔	
Training		✔	
Telephone Assistance		✔	

IL2704

Table 14. SL-GMS Software Ratings

Criteria	Rating		
	✔+	✔	✔–
Software Issues			
Standards Compatibility	✔+		
Open Architecture		✔	

IL2705-1

Table 15. SL-GMS Hardware Ratings

Criteria	Rating		
	✔+	✔	✔–
Hardware Issues			
Portability		✔	
Multiple Display Devices	✔+		
Multiple Input Devices		✔	

IL2706

Table 16. TAE Plus Capability Ratings

Criteria	Rating		
	✔+	✔	✔–
Capability Issues			
Geographical Mapping		✔	
Graphics Manipulation		✔	
Toggle Overlaps			✔–
Labels	✔+		
Text Handling	✔+		
High-Level Display Objects	✔+		
Windowing	✔+		
Code Generation		✔	
User Logging			✔–

IL2707

Table 17. TAE Plus Usability Ratings

Criteria	Rating		
	✔+	✔	✔–
Usability Issues			
Graphic Icons			✔–
Direct Manipulation			✔–
Shallow Menuing	✔+		
Object Placement		✔	
Environment Flexibility			✔–
Screen Overview			✔–
Response Time		✔	
Development Without Coding			✔–
Novice Versus Expert User Support			✔–
Learning Curve	✔+		
Usable Documentation		✔	
Confirmation Checks		✔	
File Recovery			✔–

IL2708

Table 18. TAE Plus Vendor Support Ratings

Criteria	Rating		
	✔+	✔	✔–
Vendor Support Issues			
Installation Assistance			✔–
Evaluation Period		✔	
Training		✔	
Telephone Assistance	✔+		

IL2709

Table 19. TAE Plus Software Ratings

Criteria	Rating		
	✔+	✔	✔–
Software Issues			
Standards Compatibility	✔+		
Open Architecture	✔+		

IL2710

Table 20. TAE Plus Hardware Ratings

Criteria	Rating		
	✔+	✔	✔–
Hardware Issues			
Portability	✔+		
Multiple Display Devices	✔+		
Multiple Input Devices		✔	

IL2711

Table 21. DataViews Capability Ratings

Criteria	Rating		
	✔+	✔	✔–
Capability Issues			
Geographical Mapping		✔	
Graphics Manipulation		✔	
Toggle Overlaps		✔	
Labels	✔+		
Text Handling		✔	
High-Level Display Objects	✔+		
Windowing		✔	
Code Generation			✔–
User Logging			✔–

IL2712

Table 22. DataViews Usability Ratings

Criteria	Rating		
	✔+	✔	✔–
Usability Issues			
Graphic Icons	✔+		
Direct Manipulation	✔+		
Shallow Menuing			✔–
Object Placement		✔	
Environment Flexibility		✔	
Screen Overview			✔–
Response Time		✔	
Development Without Coding	✔+		
Novice Versus Expert User Support		✔	
Learning Curve		✔	
Usable Documentation			✔–
Confirmation Checks	✔+		
File Recovery		✔	

IL2713

Table 23. DataViews Vendor Support Ratings

Criteria	Rating		
	✔+	✔	✔–
Vendor Support Issues			
Installation Assistance		✔	
Evaluation Period		✔	
Training		✔	
Telephone Assistance		✔	

IL2714

Table 24. DataViews Software Ratings

Criteria	Rating		
	✔+	✔	✔–
Software Issues			
Standards Compatibility	✔+		
Open Architecture		✔	

IL2715

Table 25. DataViews Hardware Ratings

Criteria	Rating		
	✔+	✔	✔–
Hardware Issues			
Portability		✔	
Multiple Display Devices	✔+		
Multiple Input Devices		✔	

IL2716

Table 26. SET Capability Ratings

Criteria	Rating		
	✔+	✔	✔–
Capability Issues			
Geographical Mapping		✔	
Graphics Manipulation		✔	
Toggle Overlaps			✔–
Labels		✔	
Text Handling	✔+		
High-Level Display Objects		✔	
Windowing	✔+		
Code Generation		✔	
User Logging			✔–

IL2717

Table 27. SET Usability Ratings

Criteria	Rating		
	✔+	✔	✔–
Usability Issues			
Graphic Icons			✔–
Direct Manipulation		✔	
Shallow Menuing	N/A		
Object Placement		✔	
Environment Flexibility		✔	
Screen Overview			✔–
Response Time		✔	
Development Without Coding			✔–
Novice Versus Expert User Support		✔	
Learning Curve			✔–
Usable Documentation		✔	
Confirmation Checks			✔–
File Recovery			✔–

IL2718

Table 28. SET Vendor Support Ratings

Criteria	Rating		
	✔+	✔	✔–
Vendor Support Issues			
Installation Assistance	✔+		
Evaluation Period		✔	
Training		✔	
Telephone Assistance	✔+		

IL2719

Table 29. SET Software Ratings

Criteria	Rating		
	✔+	✔	✔–
Software Issues			
Standards Compatibility	✔+		
Open Architecture			✔–

IL2720

Table 30. SET Hardware Ratings

Criteria	Rating		
	✔+	✔	✔–
Hardware Issues			
Portability		✔	
Multiple Display Devices		✔	
Multiple Input Devices		✔	

DISTRIBUTION LIST

D040:

D. I. Buckley
J. G. Koehr
J. C. Naylor, Jr.

D041:

S. W. Dardinski
G. R. LaCroix
B. M. McCay
L. E. Taylor

D042:

T. Aiken
J. S. Blackwell
S. E. Blomquist
J. R. Boonstra
C. D. Bowen
S. E. Campbell
W. E. Collins
R. G. Couture
D. L. Cuomo
M. E. Daniels
N. C. Goodwin
D. D. Gregorio
E. M. Halbfinger
L. I. Hoffberg
D. J. Kurys
E. S. Michlowitz
M. J. Murphy (10)
S. V. Offutt
A. M. V. Papia
S. B. Parrish
M. D. Patron
P. O. Perry
M. J. Reece
J. J. Shottes

D043:
R. J. Barnick, Jr.
J. Logan

D043 (continued)

M. K. Pulvermacher
B. A. Ranzinger

D044:

N. L. Anderson
S. F. Crisp
A. C. Hoffman
C. R. Triebs

D045:

J. P. Burke
J. R. Earlam
D. Karakitsos
R. S. Miller

D046:

J. J. Burke
C. F. Lowell

D047:

L. A. Dailey
T. R. Hoyt
C. L. Williams
C. W. McClure

D048:

B. A. Feldmeyer
R. L. Ryan
J. L. Taddonio

D049:

T. L. Folk
L. M. Schultz